WIRELESS NODAL SENSING FOR CIVIL ENGINEERING SYSTEMS

Todd C. Radford

B.Sc.Eng (Civil Engineering)
University of New Brunswick, Canada, 2000

Submitted to the Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Civil and Environmental Engineering
at the
Massachusetts Institute of Technology

May, 2003

Signature of Author: _____
Department of Civil & Environmental Engineering
May 9, 2003

Certified by: _____
Jerome J. Connor
Professor of Civil & Environmental Engineering
Thesis Supervisor

Accepted by: _____
Oral Buyukozturk
Professor of Civil & Environmental Engineering
Chairman, Departmental Committee on Graduate Studies

Wireless Nodal Sensing for Civil Engineering Systems

by

Todd C. Radford

Submitted to the Department of Civil and Environmental Engineering
on May 9, 2003 in Partial Fulfillment of the
requirements for the Degree of Master of Science
in Civil and Environmental Engineering

Abstract

Sensing technology has been changing rapidly in recent years. These changes have made the monitoring of physical infrastructure much more feasible. This possibility raises the argument of whether civil engineers should deal with sensors and sensing technology. This topic is discussed and it is suggested that sensing fits well into civil engineering as a link between physical and virtual infrastructure. Furthermore, civil engineers are uniquely qualified to deal with sensing systems due to their knowledge of the applications and proficiency with large-scale systems.

Considering the nature of physical infrastructure, wireless nodal sensing offers a lot of potential. MEMS technology is explored as an enabling technology that has made wireless nodal sensing possible.

A wireless sensor network comprised of MICA motes is set up and evaluated using excitation provided by a small shake table. The sensors are found to function adequately, but also have a number of severe restrictions, which limit the applicability for physical infrastructure. While the MICA motes are not a solution, they demonstrate potential for the use of wireless nodal sensing for physical infrastructure.

Thesis Supervisor: Jerome J. Connor
Title: Professor of Civil and Environmental Engineering

Table of Contents

Chapter 4 - A Wireless Nodal Sensing System

Chapter 5 - Conclusion

References

Appendices

Introduction

One of the major current global concerns is sustainability. Sustainability should be especially important to civil engineers considering the scale and importance of civil engineering works. If a piece of infrastructure fails, it both affects lives and is expensive to fix. It is therefore important to ensure that infrastructure systems are sustainable, and this is generally done by increasing the lifespan and ensuring that they can be dealt with as this lifespan reaches its end.

There are several methods of increasing infrastructure performance and lifespan. These include using improved design techniques, using high-performance materials, and making systems adaptive. Adaptive systems have not had a great deal of implementation, mostly due to lacking technology. However, recent advances in technology have made these systems more feasible.

Some of the most significant advances come in the form of monitoring systems, a necessary part of any adaptive system. One of the advances that has made monitoring cheaper and more effective is wireless nodal sensing.

This thesis will discuss sensing and monitoring in general and as they apply to civil engineering. The focus will be the application of a wireless nodal technology for infrastructure systems.

Chapter 1 will introduce sensing and how it applies to civil engineering and infrastructure. Sensing is generally not a part of a civil engineering curriculum, and this fact will be discussed in terms of reasoning and the possibility of change.

Chapter 2 will discuss important issues related specifically to infrastructure and how a wireless nodal sensor network can help alleviate these issues.

Chapter 3 will discuss an enabling technology that has allowed recent development in wireless sensor networks. Main focus will be on MEMS technology, which both decreases cost and increases functionality.

Chapter 4 will examine a simple wireless nodal sensor network and its functionality.

Chapter 5 will discuss results and some technology related to other wireless networks that may indicate how the field is likely to progress.

Chapter 1 - Sensing

1.1 - What is sensing?

Sensing encompasses methods for gathering information about a system. Sensing is performed through the use of sensors. A sensor is a device which changes an observed signal into an output, which can be interpreted and used by some sort of recording, transmitting, or analyzing system. This is purposefully vague since the sensing field is extremely broad. The mechanics of a sensor depend specifically on what is being observed and how this observation is to be used.

Information about a system gathered through sensing yields an understanding of the state of the system. The specific properties in question vary between systems, but an understanding of state can be useful for most. It is this that makes sensing an important field.

1.2 - What is physical infrastructure?

Infrastructure is defined as the basic facilities, services, and installations needed for the functioning of a community or society. Physical infrastructure more specifically refers to installations and other physically tangible systems. Buildings, bridges, roads, traffic structures, sewer systems, water distribution systems, and tunnels all fit into physical infrastructure. These are all systems that are designed and built by civil engineers.

The importance of physical infrastructure is often overlooked by the population in general. Infrastructure is a necessity for the functioning of society, and as such is often taken for granted, provided it is in working order. This is what makes

physical infrastructure different from many other fields. The public's perception of infrastructure is unique, and this has a strong influence on the owners, who are generally government and public officials. The issues that this creates are very important and will be discussed in Chapter 2.

1.3 - Why would we want sensing for physical infrastructure?

The question "why?" should be, and usually is, asked for any sort of new science or technology. In some fields, the answer is not as important, since the pursuit of the knowledge can be considered as important as the outcome. However, in a field as practical as civil engineering there needs to be a strong justification for most research, in terms of both applicability and feasibility.

Simply by looking at the definition presented above, we understand the importance of infrastructure. Infrastructure is an integral part of society and is a necessity for its proper functioning. Of course, this assumes proper functioning of the infrastructure itself. While this is something that is often overlooked by the population, it is nonetheless important. The failure of an infrastructure system can cause extreme damage to a community. It is therefore imperative that these systems do not fail. The main method of ensuring this is through proper design. However, this is only effective within a certain lifespan and should be assisted through regular monitoring and maintenance. The monitoring of a physical infrastructure system is done through the use of sensing.

The idea of monitoring infrastructure is not new. Structural health monitoring has been around for many years, though examples of practical implementation are limited. In many places throughout the world, there is aging infrastructure, which for one reason or another cannot be replaced. In these cases, maintenance and monitoring have been implemented to increase the functional life. This is only somewhat effective.

1.4 - Why should civil engineers learn about sensing?

There are two major considerations when discussing why civil engineers should deal with sensing. The first relates to how sensing is important in civil engineering systems. The second relates to why civil engineers in specific should learn about sensing. Both of these points will be investigated.

1.4.1 - Life-cycle design

Civil Engineers tend to deal with large-scale infrastructure systems. They work on almost all aspects including planning, design, construction, maintenance, and operation. The idea of life-cycle design is to consider all of these aspects from the beginning. This is not a new idea. Life-cycle design has been around for years and is incorporated in many mechanical and aerospace systems. However, the idea has not become widespread within civil engineering and physical infrastructure. There are several reasons for this.

Firstly, civil engineering systems tend to have relatively long lives. This creates a large amount of uncertainty as to what sorts of loading a system will take over its entire life. This uncertainty tends to make people hesitant to take responsibility for the entire life of the system.

Secondly, the long life creates uncertainty as to the owners of the system. In terms of public works, a politician is not likely to be in office over the entire life of a physical infrastructure system. There is therefore little incentive to spend more money initially to take into account problems that will occur when someone else will have to spend the money. It is near-sighted, but a problem nonetheless.

However, even considering these problems, civil engineering is moving in the direction of life-cycle design. By looking at the past, we can see the integration of different aspects of design. Where once there were separate architects, engineers, contractors, and owners/operators, we now see collaboration in the form of design-build/design-build-operate firms and organizations. The integration of different components of the overall process creates a more efficient system. However, there are many more aspects that must be taken into consideration in the design to realize this efficiency. Matters such as construction method and schedule, operation, and maintenance can be integrated in the original design. Life cycle design can follow from these types of systems.

There two main reasons why life cycle design should be considered as worthwhile: one being practical and one being moral. Practically, life-cycle design has the potential to make systems both more effective and more efficient. By taking matters into account at the beginning of the process, there is a greater initial cost of money and effort. However, the savings over the life of the system generally vastly outweigh this initial cost.

Morally, it is important for civil engineers to take responsibility for their systems. Public works and civil systems tend to get ignored in terms of maintenance and rehabilitation until it is almost too late. Furthermore, physical infrastructure systems have a huge impact on people and society, and can cause dramatic impact in the case of failure. By making civil systems more efficient and effective, and by increasing their life through proper design and maintenance, we help society in general. On an environmental note, taking into consideration the end of the system, its destruction, and the use or disposal of the materials makes a lot of sense.

1.4.1.1 - What does life cycle design have to do with sensing?

To implement a complete life cycle design, one of the aspects that must be considered is maintenance and rehabilitation. Both of these can be made more effective and efficient through the use of a sensing or monitoring system. To assess when and where to maintain, it is useful to know the state of the system. The state of any system can be established by observation, which is performed with the use of a sensing system. There has been a large amount of work done in the area of structural health monitoring, which relates closely to this goal, but there is much work that needs to be done.

Sensing and monitoring systems can also be incorporated into the operation of a system. By monitoring for extreme events or effects, control systems can be implemented which allow reduction in construction materials. For example, a system designed to monitor inter-storey drift in a building for damage assessment could be used in a feedback loop to activate dampers or stiffeners to help control motion.

1.4.2 Why should civil engineers deal with sensing?

Sensing and monitoring systems involve a large variety of technology and knowledge. It is a broad field that doesn't fit into the traditional concept of civil engineering. For this reason, many civil engineers feel that it is best left to electrical and mechanical engineers. However, there are several compelling reasons for civil engineers to deal with sensing.

Firstly, sensing can be used to contribute to the performance of civil engineering systems. However, civil engineering systems tend to be different from other systems. Therefore, to design sensing systems and use technology for civil applications requires specific knowledge. This makes the civil engineer uniquely

qualified to work with sensors since they have a strong knowledge of the application.

More generically, civil engineers tend to deal with large-scale distributed systems. Sensing systems, especially ones for civil applications, tend to have large numbers of components and cover large distances. While the technology is different, there are many similarities between sensing systems and civil applications.

Civil engineering is often considered to be an unchanging field. However, it is important that the field remain dynamic by incorporating new technology into existing civil applications and generating new applications for the technology. The application of sensing in general is not a field that has been taken over by anyone in specific. It is a great opportunity for civil engineers, who have strengths in systems engineering, to expand the field in a way that will benefit existing civil applications and provide new areas of work.

1.5 - How does sensing fit in to a civil engineering framework?

Civil engineering is a field that can be very difficult to describe. Above we defined infrastructure as the backbone of society, comprised of all the things that make a society function. Physical infrastructure is the area that is generally associated with civil engineers. However, the world is evolving and civil engineering must evolve as well.

The Department of Civil and Environmental Engineering at MIT is an example of such an evolution. Civil engineering at MIT has always had a very theoretical approach to civil engineering, which is by its nature a practical field. Emphasis tends to be placed more on modelling than on experimentation and the facilities are designed as such. Furthermore, by being part of such a forward thinking

technical institute, the civil engineering department has tried to keep at the front of the field while steadily losing students to the more technology-based fields such as electrical engineering and computer science.

One of the interesting aspects of the civil department is the presence of an Information Technology program for graduate students. This seems odd if one considers a traditional definition of civil engineering. Furthermore, the IT program does not focus specifically on IT for civil engineering systems, but on IT in general and how it can be applied to a variety of systems. This doesn't fit within a traditional civil engineering framework. Does this mean we should exclude IT from civil engineering, or change our view of the framework?

The author would like to suggest that civil engineering deals with infrastructure. We can fit bridges and buildings into this as physical infrastructure. Perhaps we can view IT as virtual infrastructure. IT deals with the functioning of systems. Many systems in our modern age cannot function without information technology, and therefore to deal with it seems entirely within the realm of civil engineering.

If we take civil engineering as encompassing both physical infrastructure and virtual infrastructure, there needs to be a link between the two. This comes in the form of sensing. In order to interact with physical infrastructure, we need to have information about the system. Sensing and monitoring systems provide this information. This information is then processed, manipulated, and used by the virtual infrastructure. This makes up the components of a fully integrated system, with sensing as an important part.

Chapter 2 – Major Issues

2.1 - Overview

There are significant constraints within any field that make its problems unique. Physical infrastructure and civil works are no exception. It is important to understand these constraints, and the underlying issues, and how they affect projects before working within this framework.

2.2 - Characteristics of civil engineering systems

There are several characteristics that define civil engineering systems. Individually, the characteristics are applicable to systems in other fields, but as a whole they make civil systems unique.

The first is scale. Physical infrastructure tends to be large scale, both in terms of the number of components and the physical size. There are very few other fields that deal with systems of as large a scale, telecommunications being a significant exception.

Secondly, physical infrastructure has a relatively long life. This means that not only must it be designed to survive under existing loads, but the changes in loading over the life of the system must be considered.

Thirdly, there is a certain inaccessibility to physical infrastructure. The term "inaccessibility" is used to cover a number of different problems. First, civil systems can be in remote or inaccessible locations. Second, it is not generally a simple process to dismantle a civil system apart in order to enact repairs. And

finally, the civil system is static and cannot be removed to another location or from environmental conditions.

Fourthly, civil systems are almost all unique. Due to the varying loads and requirements of different locations, it is essentially impossible to implement an identical system in two different locations.

Finally, the economics of civil systems must always be considered. One of the most significant constraints for most applications is economic. However, the nature of civil engineering exarcebates this problem. Civil engineering works are often publicly funded. Public funding generally means that there are a lot more restrictions on spending. Also, money is controlled by politicians, who are often more concerned with short-term effects than long-term benefits. This type of short-sighted development has caused many problems.

These are the main aspects that must be considered in the development of any sort of sensing system. It is important to consider what qualities would be useful for a sensing system to have in order to overcome some of these issues.

2.3 - Sensing System Requirements

2.3.1 - Low-Cost

Reducing cost is an important consideration in the design of most systems, however it is especially important in civil applications for several reasons. First is the economic situation for both public and private works. Sensing systems tend to add safety and increase functional life, but, in most cases, do not add significantly to functionality. This can make the justification of cost difficult. It is therefore important to keep overall system costs as small percentages of overall project costs. Furthermore, the large scale nature of physical infrastructure

means that individual component cost must be low such that the large number of components will result in a low overall cost.

2.3.2 - Flexibility

The uniqueness of civil engineering systems eliminates the possibility of mass manufacturing, which is one of the main methods of dropping cost. A system that can be used in a variety of situations becomes critical. It must be flexible enough to be applied in a variety of situations, or it will not be feasible for broad application.

2.3.3 - Longevity

Because of the long life of physical infrastructure systems, it is important that a related sensing system will have an equivalently long life. While it is not required that the sensing system life be as long as that of the infrastructure itself, it should be a non-trivial fraction thereof. The exact fraction depends on the specific system, and must be determined based on a number of considerations including the relative longevities and the replacement cost of the sensing system.

2.3.4 - Reliability and Low Maintenance

Both contributing to, and because of, the required longevity of sensing systems, it is important that they are reliable and require low maintenance. Reliability and low maintenance are often related, but not implicitly. A reliable system will continue to both function and function properly, without maintenance. Maintenance not only consists of correcting for malfunctions, but also regular upkeep such as battery replacement and recalibration. Therefore to achieve low maintenance it is important to have a system that yields consistent results and has either a long-lasting or self-replenishing power source.

2.4 – Potential Solutions

It is desirable, and perhaps even necessary, to find a type of sensing system that has these qualities in order to make monitoring physical infrastructure feasible for practical implementation. There are a number of different possibilities, but one that offers a great deal of potential is a system of wireless nodal sensors.

Wireless nodal sensors, or network embedded sensors, offer potential for use with physical infrastructure due to their inherent flexibility. A network embedded sensor is considered to be a nodal element that consists of at least a sensing element, some sort of microprocessor, and a method of wireless data transfer. This results in a self-contained sensor that is connected to other sensors and an overall data collection or control system through a wireless communication network. This wireless network allows sensors to be placed in any arrangement that is required. This flexibility fits well into our requirements for physical infrastructure.

However, flexibility is not the only requirement. Low cost is also incredibly important. Currently, a single nodal wireless sensor is not less expensive than an equivalent wired sensor. However, the cost that is saved in wiring and installing the system, combined with decreasing cost of microprocessors and radio modules defrays this difference. The cost of a wireless system is still greater, but, if current trends continue, it is predicted that this will change.

Also, there is some question as to the reliability and longevity of these systems. Longevity can only be truly proven through long-term application, however, it is possible to test the reliability. There are some quantitative methods of assessing the maintenance level, however, such as examining the power consumption, which relates to the frequency of battery replacement.

There are a number of enabling technologies that make wireless nodal sensor networks more feasible. These are discussed in more detail in Chapter 3.

A sample wireless nodal sensor network is presented and examined in Chapter 4.

Chapter 3 - Technology

3.1 – Overview

To be able to design and implement an effective sensing and monitoring system it is imperative to have a strong understanding of the technology involved. Specifically, there are several technologies that have changed the market and made sensing for civil engineering applications more possible. These are termed enabling technologies, and are explained first. This is followed by sensor and sensing technology with a focus on physical measurands. Finally, related sensor system technology is examined.

3.2 - Enabling Technology

An enabling technology for a specific application is any advancement that overcomes or helps overcome some restriction or problem that is stopping the application from proceeding. As discussed in Chapter 2, the significant issues that restrict the application of sensing systems for physical infrastructure are economics and physical limitations. The enabling technology that we look at is MEMS technology, which offers decreased cost and improved functionality.

3.2.1 - MEMS

One of the main driving forces behind this investigation into sensing technology is the MEMS technology. MEMS, or Microelectromechanical Systems, are micro-scale devices, fabricated using microfabrication techniques, which integrate electronic and mechanical capabilities. MEMS developed out of the microelectronics industry, but has since become a strong separate industry. The main trademarks of MEMS are the microfabrication procedures. It is these

techniques that allow for decreased cost through batch fabrication, which makes MEMS technology an enabling technology and a driving force for this research.

The most important material in MEMS, as with microelectronics, is silicon. Silicon is an important material because it has both interesting electrical and mechanical properties. Electrically, silicon is a semiconductor that can be doped with positive or negative ions to change its properties. Doped silicon forms the basis for many of our current electronic devices. Structurally, silicon offers a relatively high strength and Young's modulus. However, more important is the fact that the mechanical properties are relatively consistent, making the behaviour predictable.

3.2.1.1 - Microfabrication

There are a number of microfabrication techniques that have associated advantages and disadvantages. The main techniques will be described and compared to gain knowledge of abilities and limitations. The three most significant techniques are bulk micromachining, surface micromachining, and LIGA. More information on microfabrication can be found in Madou (1997.)

The first step in many microfabrication procedures is the creation of a mask, which is used to define the shape of the structure that is to be created. A mask is essentially a stencil of the desired shape. By using a photoresist, a material that changes property under ultraviolet light, and a UV source, the mask pattern can be transferred to the material that will be shaped. By using selective material removal and deposition techniques and resist stripping, a wide variety of shaping procedures become available.

3.2.1.1.1 - Bulk Micromachining

Bulk micromaching is a subtractive process with the defining characteristic that material is removed directly from the silicon substrate. There are two distinct types of bulk micromachining: dry and wet chemical.

Dry bulk micromachining includes a number of different procedures, the most common of which is deep reactive ion etching or DRIE. All of the procedures involve the etching of the solid-state surface in the gas phase. DRIE uses both physical etching through ion bombardment and chemical etching through chemical reaction. This combination allows the creation of structures with high aspect ratios.

Wet chemical micromachining uses chemical etchants in liquid phase to remove base material. Isotropic etchants, such as nitric acid, remove material uniformly in all directions. Anisotropic etchants, such as KOH, have different rates of removal determined by the crystal planes of the substrate. This property is used to create smooth surfaces at specific angles.

Bulk micromaching often takes advantage of anodic bonding to permanently attach two silicon wafers. This allows for the creation of a wider variety of structures.

3.2.1.1.2 - Surface Micromachining

Whereas bulk micromachining is a subtractive process that acts on the silicon wafer, surface micromachining is an additive process that uses the silicon wafer simply as a base. Surface micromachining is generally characterized by the deposition and etching of thin films on the silicon wafer. Though surface micromachining can involve a large number of layers and materials, the basic

process involves a structural material that will form the final shape and a sacrificial material that is used in the fabrication and subsequently removed to release the structural material. The most common structural material in surface micromachining is polysilicon.

The basic process in surface micromachining starts with the deposition of a sacrificial material on the silicon wafer. This material is then etched using a first mask. This pattern generally represents the base of the final structure. The structural material is then deposited over the sacrificial material, filling in the voids where the sacrificial layer was etched. The structural layer is then etched using a second mask. This creates what is essentially the final shape of the structure. Finally, the sacrificial material is removed to release the structure.

One of the most important topics in surface microfabrication is the determination of the mechanical properties of thin films. Thin films can have vastly different properties from the same material in bulk form. These properties can be affected by the deposition method, deposition properties such as rate, temperature, and pressure, and thickness. In addition to changing basic properties, the above can also generate internal stresses, which can greatly affect behaviour. It is important to understand how the thin film materials will behave in order to design a surface micromachining process properly.

3.2.1.1.3 - LIGA

LIGA stands for lithography, electroforming, and moulding in German, where it was first implemented. LIGA uses x-ray lithography and electroforming to create structures with extremely high-aspect ratios. This overcomes one of the major problems with bulk and surface micromachining. In addition, the accuracy and resolution available through LIGA is incredibly high. However, the resolution is higher than is generally necessary for MEMS applications. Bulk and surface

micromachining offer adequate performance and are much more cost-effective. However, LIGA still remains the best established method of microfabrication and offers opportunities for the creation of novel three dimensional structures. LIGA can also become cost effective if used in the creation of moulds that can then be used to generate large numbers of products.

The main material used in LIGA is nickel, although other metals that are compatible with electrodeposition can be used. After x-ray lithography has been completed to obtain the structure, nickel can then be electroplated onto the resist structure. This can be the final product, or can be used to create a permanent mould for the creation of more products.

3.3 - Sensors

A sensor may be defined as a device that converts a non-electrical quantity into an electrical signal (Gardner 1994.) Generally, much must be done to this electrical signal before it becomes useful, however the sensor remains the key starting point for any sort of measurement. To implement a monitoring system, it is important to have a good grasp of the functionality of the sensors involved, which is best obtained through an understanding of the underlying principles.

Focus will be given primarily to sensors with measurands that are applicable to physical infrastructure applications. This document will focus on mechanical sensors.

3.3.1 - Mechanical Sensors

There are a number of mechanical properties that can be measured. The most significant for physical infrastructure applications is motion. Generally, we are interested in position and its first and second derivative with respect to time.

Displacement, velocity, and acceleration can all be measured with differing levels of difficulty, accuracy, and usefulness. Acceleration will be examined first as it is the most commonly measured mechanical measurand.

3.3.1.1 - Acceleration Measurement

Acceleration is very important in civil engineering. Accelerations, including gravity, generate forces proportional to the masses on which they are acting. These forces must always be taken into account in design. People and machinery are also generally very sensitive to acceleration. Furthermore, acceleration is currently used as a basis for determining velocity and displacement.

The basic equation of equilibrium for a dynamic system is

$$mx''+cx'+kx=P(t)$$

where m is mass, c is damping, k is stiffness, x is position in time, and P(t) is loading as a function of time. In its simplest form, an accelerometer can be considered as a basic second order system, and behaves with the dynamic properties of such a system. When the proof mass is subjected to acceleration, an inertial force relative to the size of the mass is generated. This force is carried by an attached spring of stiffness k. The displacement of the spring and the force in the spring are thus both proportional to the acceleration applied to the proof mass. This is the basic operation of an accelerometer and is valid for a quasi-static system (ie when x''' is small.) However, things become more complicated by damping when the system is dynamic. However, the main measurement is still the displacement of the proof mass. See Chopra (1995) for a more detailed explanation of dynamic properties for SDOF systems.

3.3.1.2 - Types of Accelerometers

Though all rely on the basic concept of a moving proof mass, there are a number of different methods of measuring mass motion. Most current accelerometers are MEMS devices that are built using microfabrication procedures. While traditional accelerometers can be obtained, the low-cost, small size, and improved performance make microdevices a better choice for most applications. For this reason, we will examine primarily microaccelerometers.

3.3.1.2.1 - Capacitive-based Measurement

In an electrical system, electromotive force pushes electrons through conductors. A capacitor is a device that can hold charge when it is subjected to an EMF. The capacitance is the capacity of charge that can be stored and is based on the physical properties of the capacitor.

There are several types of capacitors, but we will focus here on only one: the parallel plate capacitor. A parallel plate capacitor has two plates that are held separate, with air or some other material between. The capacitance of the system is based on the overlapping area, the type of materials used, and the distance between the plates.

Since the gap between the plates is such an important factor in capacitance, it is obvious how this can be used in an accelerometer. The ability to measure the capacitance can tell us the distance between the plates, assuming everything else remains constant. Given this distance, we can then establish the acceleration based on the dynamic properties of the system.

The use of capacitance to measure acceleration is not a simple problem. Firstly, the measure of capacitance is not straightforward. Secondly, the relationship

between gap and capacitance is inherently non-linear. Thirdly, fringe effects in the electric field generated in the capacitor create problems and generally lead to assumptions about the behaviour. Finally, there is a large amount of computation necessary to change the electric output into useful results. Furthermore, capacitive measurement is susceptible to electromagnetic interference.

However, capacitive measurement has a number of advantages in that it is not sensitive to temperature in the same way that more material property-based sensing methods are, nor is it as susceptible to residual strains from the fabrication process.

Capacitive accelerometers also allow for some flexibility of use. Capacitance is a function of a number of factors, including displacement as previously mentioned. However, the electromagnetic field is also strongly affected by the overlapping area of the surfaces. This means that we can not only use normal, but also transverse motion of the surface to measure accelerations. Figure 3.1 shows a capacitive microaccelerometer with a torsional suspension, which uses overlapping area as the varying parameter.

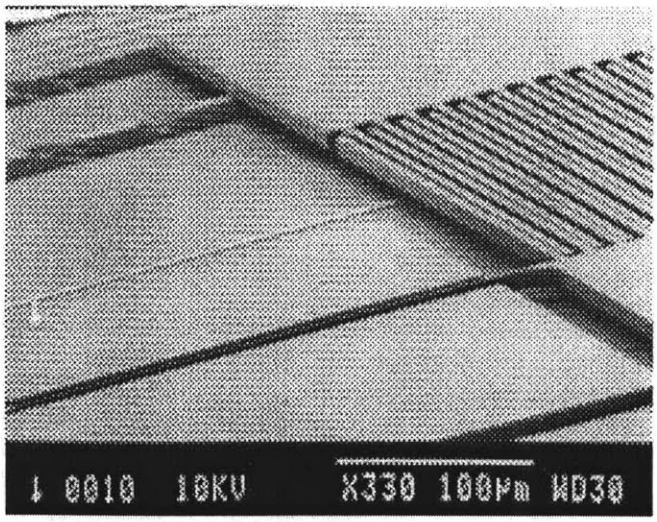


Figure 3.1: Combed capacitive accelerometer (Selvkumar and Najafi, 1998)

Capacitive measurement can also be increased in strength and sensitivity by using a combed structure, as shown in Figure 3.1. Non-touching interlocking teeth provide a much stronger electromagnetic field than a flat surface of equivalent area.

3.3.1.2.2 - Piezoresistive-based measurement

Piezoresistivity is a property by which the resistance of a material changes with the application of a strain. This is a result of the crystal structure of the system.

A piezoresistive material can be used very effectively in a variety of sensing mechanisms. Conditioning electronics, generally in the form of a Wheatstone bridge, can measure the resistance of the material, which can be translated into strain. Through force balance, the strain can be linked to the displacement of the proof mass.

Piezoresistive accelerometers are currently the most popular and prevalent commercial accelerometers, used for such applications as automotive airbags. Silicon can be doped to become a piezoresitive material. Using silicon as the base structural material makes it relatively straightforward to measure the strain in the support material. As shown in Figure 3.2, the supporting can be made of piezoresistive material, combining the sensing with the structure of the sensor.

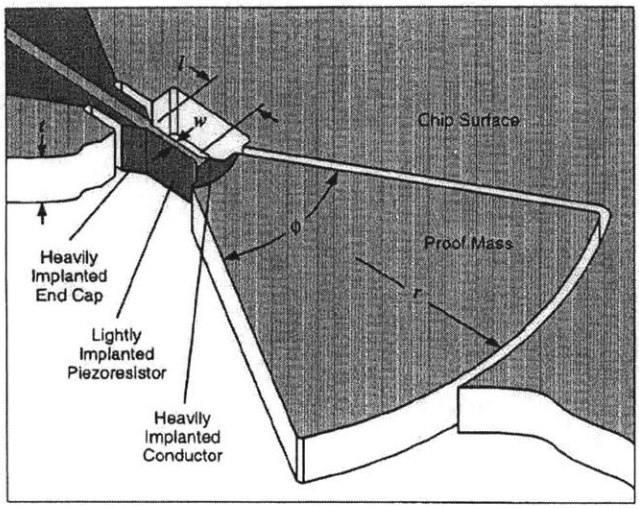


Figure 3.2: Schematic of a piezoresistive accelerometer (Partridge et al, 2000)

3.3.1.2.3 - Piezoelectric-based measurement

Piezoelectric accelerometers function in generally the same manner as a piezoresistive accelerometer. A piezoelectric material is one in which a voltage is generated under an applied mechanical strain and a mechanical strain is generated under an applied voltage. Piezoelectric properties arise from crystal structure, somewhat similar to piezoresistive. The main example of piezoelectric materials is ZnO, though there are many varieties that can be used.

Figure 3.3 shows two sample structural schematics for piezoelectric accelerometers.

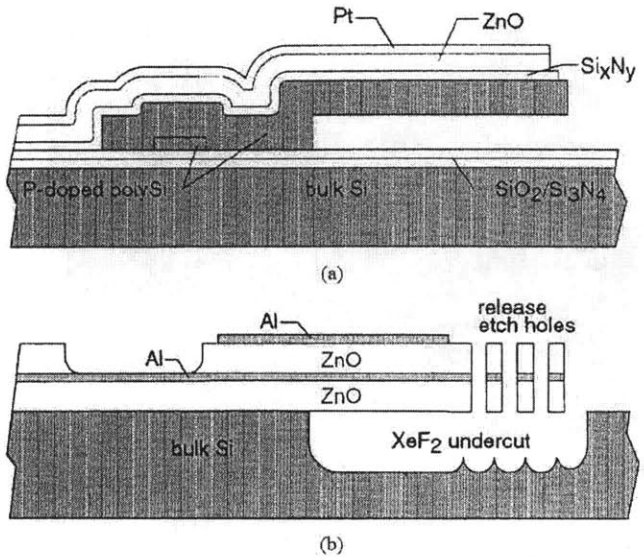


Figure 3.3: Two sample structural schematics for a piezoelectric accelerometer (DeVoe and Pisano, 2001)

3.3.1.2.4 - Other Measurement Techniques

3.3.1.2.4.1 - Resonant Frequency

The resonant frequency of a member is dependent on the tension applied through that member. This can be shown most clearly by a guitar string, which can be tuned by applying more or less tension through a turned key. The property of tension-dependent frequency can be used as a measurement technique. By attaching a secondary non-structural member to the proof mass in an accelerometer, the movement of the mass can be measured by detecting the changes in natural frequency of this member. This requires some sort of excitation to be applied to the member in order to initiate resonance. This excitation is generally temperature-related. Figure 3.4 shows a schematic of a possible layout for a resonant frequency-based accelerometer.

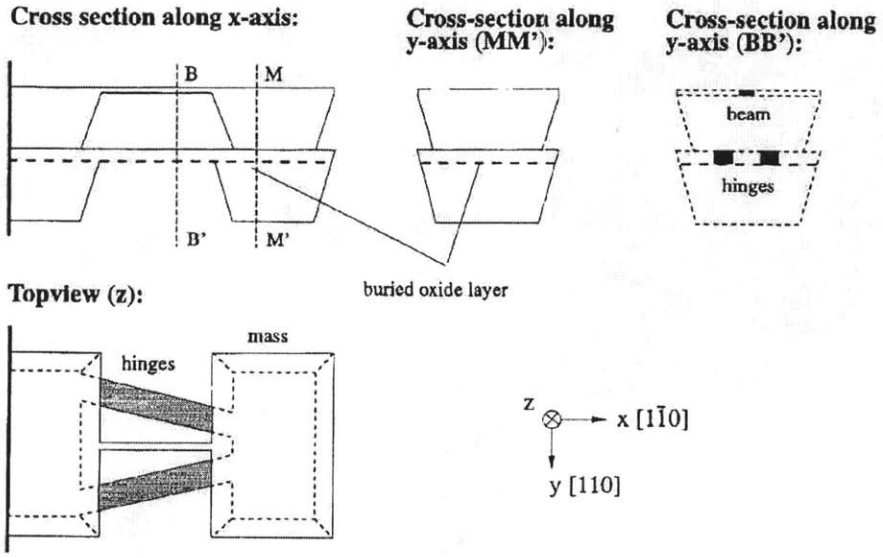


Figure 3.4: Sample schematic for resonant frequency-based accelerometer (Burrer et al, 1996)

3.3.1.2.4.2 - Electron Tunneling

When capacitive plates are brought sufficiently close together, a physical phenomenon called electron tunneling will occur. With sufficient proximity, electrons will pass through the dielectric, in a manner known as tunneling, creating a current through the capacitor. This process is highly dependent on the distance between the plates, and as such provides the possibility for very high sensitivities. With electron tunneling, accelerometers have been created with the ability to measure accelerations on the order of micro-g. However, electron tunneling accelerometers are still at the research level and have not yet been developed for practical implementation. Figure 3.5 shows a sample structural schematic for an electron tunneling accelerometer.

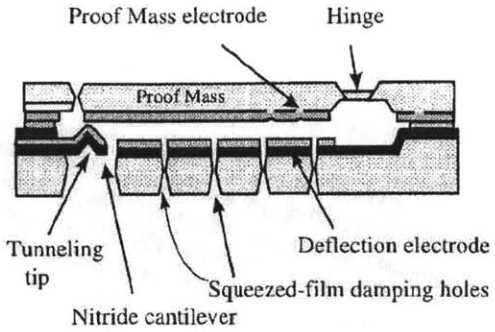


Figure 3.5: Sample schematic for an electron tunneling accelerometer (Liu et al, 1998)

3.3.1.2.4.3 - Force Balance

A force balance accelerometer requires two physical components, a sensing component and a forcing component. As the accelerometer is loaded, the sensing component detects movement. This movement passes through a feedback system, which activates the forcing component with enough force to move the sensing element back to the zero position. The force required to do this is recorded with time. Force balance is not, however, a sensing method in itself as it still requires a physical implementation. Force balance is often used with capacitive-based systems, as they can provide both sensing and force actuation.

The advantages of implementing a force balance system include the potential for improved sensitivity and the elimination of physical range limits. The disadvantages include the need for a relatively complex control/feedback system and a more complex structure to the sensor itself.

3.3.1.2 - Velocity and Displacement

While it would be very useful to be able to measure velocity and displacement in addition to acceleration, there are not currently any technologies that allow this

to be done nodally. Acceleration is possible to measure since it can be considered to be an absolute measure. For civil applications, constant acceleration is essentially impossible, or irrelevant, therefore it is a simple matter to define zero acceleration. This is not the case for velocity and displacement.

Both velocity and displacement must be measured in a relative manner. There are a number of techniques for measuring distance and relative speed, however these all rely on a physical reference point. While these sensing techniques can be used in some situations, their physical constraints create significant restrictions. While there is some potential for cumulative relative measurement, it will not be discussed here.

Chapter 4 - A Wireless Nodal Sensing System

4.1 - Overview

A wireless nodal sensing system could be a solution for providing sensing for physical infrastructure systems. What we would like to do is investigate the functionality of such a system and assess its potential for use in physical infrastructure.

4.2 - MICA Motes

4.2.1 - Background

The development of motes is the product of work done at UC Berkeley to create wireless sensor networks. Table 4.1 shows several generations of motes developed through this work. While there is work being done towards the next generation of mote, called Spec, it is not yet ready for implementation. The current level of mote is called MICA, a name coined from the fact that the boards are layered similar to the mineral of the same name.

For this work, a set of MICA motes is used. The motes come packaged as a set for use in research of network embedded sensors from crossbow technology, incorporated. While the MICA package is no longer available, a set of MICA2 motes, which have very similar functionality, is available. The set, which essentially consists of two sensor motes and a base station retails for $895.

Table 4.1 - The family of TinyOS motes (Gay et al, 2002)

Mote Type	WeC	rene2	rene2	dot	mica
Date	9/99	10/00	6/01	8/01	2/02
Microcontroller					
Type	AT90LS8535		ATMega163		ATMega103
Prog. mem. (KB)	8		16		128
RAM (KB)	0.5		1		4
Default Power Source					
Size	CR2450	2xAA		CR2032	2xAA
Capacity (mAh)	575	2850		225	2850
Communication					
Radio	RFM TR1000				
Rate (kbps)	10	10	10	10	10/40
Modulation type	OOK				OOK/ASK

4.2.2 - Description

The MICA motes have three major components: the sensor board, the processing board, and the power supply. A mote with the components indicated is shown in figure 4.1 and the data sheets for the mote are attached in Appendix B.

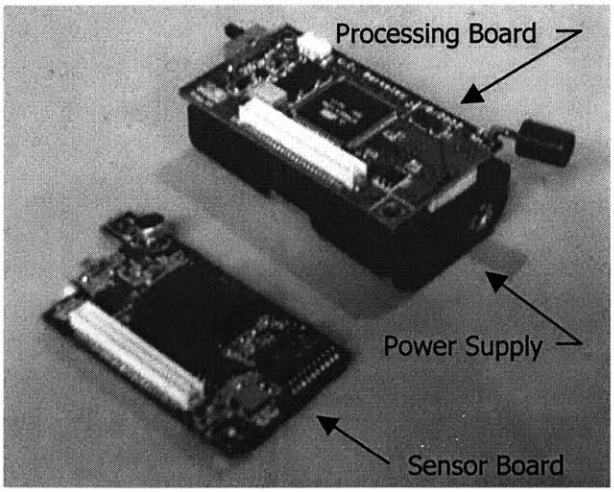


Figure 4.1: MICA Mote

4.2.2.1 - Components

Sensor board

As the motes are designed for research purposes in all aspects of operation, the sensor board is designed to offer a number of different sensing mechanisms. The processing board for the mote to be used is of the MTS310CA type that includes a photo diode, thermistor, microphone, sounder, and a magnetic and acceleration sensor. For this work, only the accelerometer will be used.

The accelerometer is an Analog Devices model ADXL202. The data sheet is attached in Appendix B. The MEMS-based polysilicon accelerometer uses differential capacitance to measure deflection, which translates to acceleration as discussed in section 3.3.1.1. The ADXL202 allows measurement of acceleration to ±2g at a sensitivity of 312 mV/g.

Processing board

The processing board for the mote to be used is of the MPR300CB type that includes an Atmel ATmega 128L microcontroller running at 4 MHz with a 10-bit 8-channel analog to digital converter. The board has a single channel, 916 MHz radio from RF Monolithics to provide bi-directional communication at 40 kbs (Gay et al, 2000.)

Power supply

Two AA batteries, mounted in a plastic battery pack, along with a DC boost converter, supply power for the motes.

Connections

The sensor and processing boards are connected together using a surface mount 51-pin connector that supports analog inputs, 12C, SPI, UART, and a multiplexed

address/data bus. The battery pack is hard-wired to the processing board.

4.2.2.2 - TinyOS

TinyOS is an operating system intended for use with network embedded sensors. The operating system was designed to function with limited resources and to take up a small amount of space in memory. To do this it provides a component-based architecture, a simple event-based concurrency model, and split-phase operations.

Component-based architecture

Part of TinyOS is a set of reusable components which can customized for specific applications. There are two types of components, configuration components and module components. Module components contain the application code. Configuration components are used to wire module components together.

Event-based concurrency

There are two types of events: tasks and events. The two are essentially the same, except that tasks are posted and will be executed when the scheduled executes the task, whereas events will preempt other tasks. Ultimately, the operation of an application is driven by hardware operation events.

Split-phase operation

Since there is no preemption for tasks, all long tasks are run as split-phase operations. This means that a single task implements an operation, while a separate task is called when the operation is complete. This prevents long tasks that cannot be preempted from monopolizing processing time.

4.2.2.3 - nesC

nesC stands for network embedded systems C. It is a programming language based on traditional C, but specifically designed for use with TinyOS and network embedded systems, as the name implies. The language has been designed to meet a number of unique challenges including the event-driven nature of TinyOS, the limited resources available, and the need for application reliability. A detailed discussion of the design of nesC can be found in Gay et al (2000.)

There are several basic principles that underlie the design of the nesC programming language. nesC is an extension of the C programming language, which provides much of the basic functionality and has the added bonus of being familiar to many programmers. There are some significant differences from C, however. First, nesC design does not provide for separate compilation. This is partially due to safety and performance issues, but mostly due to space considerations. Second, nesC is a static language and does not provide for dynamic memory allocation. Finally, nesC supports and reflects TinyOS by being similarly based around the concept of components and event-based concurrency.

The application components of TinyOS are written and compiled in nesC. This makes it a simple matter to customize the prepackaged applications that come with TinyOS for new uses. The code is then compiled and installed in the Flash memory of the nodes.

4.3 - Experimental Program

To assess the performance we would like to apply known excitation to the mote and look at the response. The input excitation will be of varying amplitude, frequency, and regularity.

4.3.1 - Setup

The experimental setup is as shown in Figure 4.2.

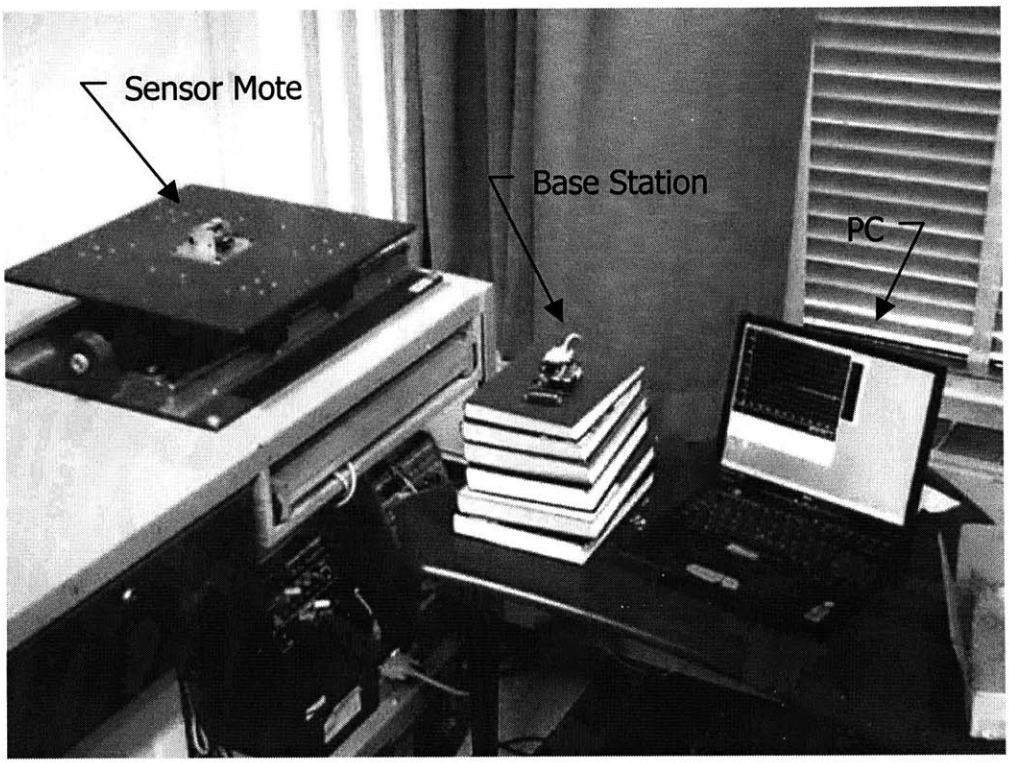


Figure 4.2: Experimental Setup

4.3.1.1 - Shake Table

Excitation is supplied by a shaking table. The shaking table to be used is the Shake Table II from Quanser Consulting (Advanced Teaching Systems.) Quanser Consulting provides products and solutions for teaching and research in control. The intended use of the Shake Table II is to create earthquake-type motions and evaluate the performance of an active mass damper, which is included with the table. For the purpose of this work, the active mass damper has been removed and the table will only be used as a source of controlled excitation.

The specific information for Shake Table II are shown in Table 4.2.

Table 4.2: Specifications for Shake Table II (Quanser Consulting Incorporated)

Parameter	**Value**	**Units**
Table dimensions	18 x 18	in
Maximum payload	33	lb
Operational bandwidth	20	Hz
Peak velocity	33	in/sec
Ball screw efficiency	90	%
Maximum force	700	N
Peak acceleration	2.5	g
Stroke	+/- 3	in
Weight	60	lb
Encoder / leadscrew resolution	.000125	in
Motor maximum torque	1.65	N-m
Ball nut dynamic loading capacity	12000	N
Ball nut life expectancy at full load	25e9	in
Linear bearing life expectancy	25e7	in
Linear bearing load carrying capacity capability	290	lb

The shaking table is equipped with #10 screw-holes located at 3.25 inches o/c. These are used as the method of attachment for a mounting bracket designed to hold a MICA mote. A photo of the mounting bracket holding a MICA mote is shown in Figure 4.3 and dimensions and specifications are shown in Figure 4.4. The mounting bracket has vertical walls to support the battery portion of the mote in the x-direction, with screws to restrict motion out-of-plane. A top bracket with screw is used to support the sensing board by providing friction between the boards and the battery module. The processing board of the mote is attached to the battery by two leads and a piece of two-sided tape. There is some worry that this connection will not be significant enough to take the loads

which are to be applied, and it is hoped that the design of the mounting bracket will account for this potential problem.

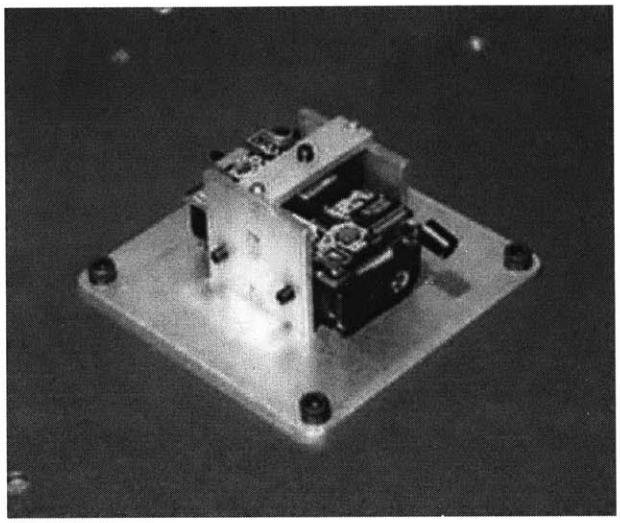


Figure 4.3: MICA mote in mounting bracket

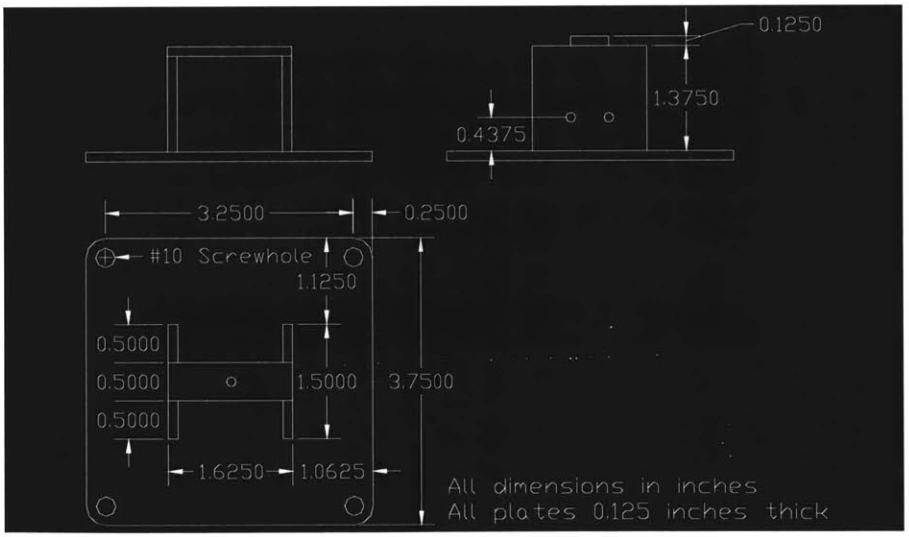


Figure 4.4: Specifications for mounting bracket

4.3.1.2 - Nodal Network

The objective of the experimental work is to assess the performance of the motes to a variety of excitations. As such, the nodal network will be kept as

simple as possible to minimize the introduction of errors. The experimental network will consist of a single sensor mote connected to the source of excitation and another single mote acting as a base station to receive collected information and pass it on to a PC, which can then be connected to a network if required. This setup will allow the evaluation of a mote's capability to detect motion and communicate this information wirelessly. The setup is also expandable and will be used as a base component in demos of wireless networks and control systems.

As shown in Figure 4.2 there are three major components to the network: the sensing mote, the base station, and the PC.

4.3.1.2.1 - The Sensing Mote

The function of the sensing mote is to measure the excitation delivered by the shaking table and transmit this information wirelessly. As such, the sensing mote is mounted to the shaking table using the mounting bracket described previously, and installed with an application that accesses the accelerometer and the RF module.

The application that will be installed on the mote is based on the Oscilloscope application that comes with TinyOS. However, there have been some major additions and changes to the original code, including the addition of a data filter. Both the module and the component code for the new Oscilloscope application are included in Appendix A.1.

4.3.1.2.1.1 - Changes to Application Code

Data source to accelerometer

The oscilloscope application originally accessed the photo sensor as the data

source. This was changed to access the accelerometer. In particular, the x-axis acceleration is used as it was found to be in better calibration than the y-axis in work from Cheekiralla et al (2002). If it is desired to use information from the Photo sensor, the #define use_accel line of code should be commented out.

Clock to Timer

The Timer function is used instead of the Clock function. This allows a more general setting of time and also allows multiple processes to use different timers. The current timer is set to tick at a rate of 50 ticks/second which defines the sampling rate of the accelerometer.

Addition of filter

The major change to the code is the addition of cubic B-spline filter. Filter_data is added as a subroutine, which is accessed every time the ADC gives notice that a piece of data is available. A description of a filter can be found at Strang and Nguyen (1997) and the application contains a number of circular buffers, which are used for the filtering. The default effect from the filtering is that velocity data is sent out delayed by 50 sampling periods.

The filtering can be eliminated from the process by commenting out
#define filter_data

Acceleration can be sent out instead of velocity by commenting out
#define send_accel

4.3.1.2.2 - The Base Station

The function of the base station is to act as a bridge between the sensor network and the PC. As such, the base station will be linked to the PC interface board attached to the serial port of the PC and will be installed with an application that

receives RF transmissions and passes them to the serial port of the computer.

The application that will be installed on the mote is the GenericBase application that comes with TinyOS. The module code, the component code, and the README file for the GenericBase application are included in Appendix A.2.

4.3.1.2.3 - The PC

The PC is used to display and record data received by the base station. There are three main applications that are used for this purpose: ListenRaw, SerialForward, and Oscilloscope.

ListenRaw

The code for ListenRaw is included in Appendix A.3, and has been modified from the original version that comes with TinyOS. The application has been modified to time stamp and to convert the data from hexadecimal to decimal. As ListenRaw can be piped into a file, it is used as the primary method of data collection from the sensors.

SerialForward

The SerialForward application is used as supplied with TinyOS. The application acts to retrieve any packets that appear at the serial port of the computer and make them available on a TCP port. This allows multiple clients to access data at the same time.

Oscilloscope

The oscilloscope application is used as supplied with TinyOS. The application allows the visualization of data retrieved in packets from the base station. The oscilloscope application acts as a client from the SerialForward application.

4.3.2 - Excitations

A series of excitations are applied to assess the performance of the network. There are four sets of excitations that are applied as different types of loading.

In general, time scales are consistent in length but are not synchronized. Therefore, the start of the waveform will be used as a visual reference to compare the input and output signals, as opposed to using the time values.

4.3.2.1 - Calibration

The first set of three excitations involves a quasi-static rotation of the sensing mote. This essentially creates a single sine wave with amplitudes at +/- 1g. As can be seen in Figure 4.5, the output of the accelerometer is in no set units and must be scaled accordingly. For the rest of the tests, all output will be scaled by 1/60 g/unit. This test is also used to verify the performance of the mote for static accelerations. As can be seen from the test, the mote works very well at recording static accelerations, and as such would work very well as a tiltmeter.

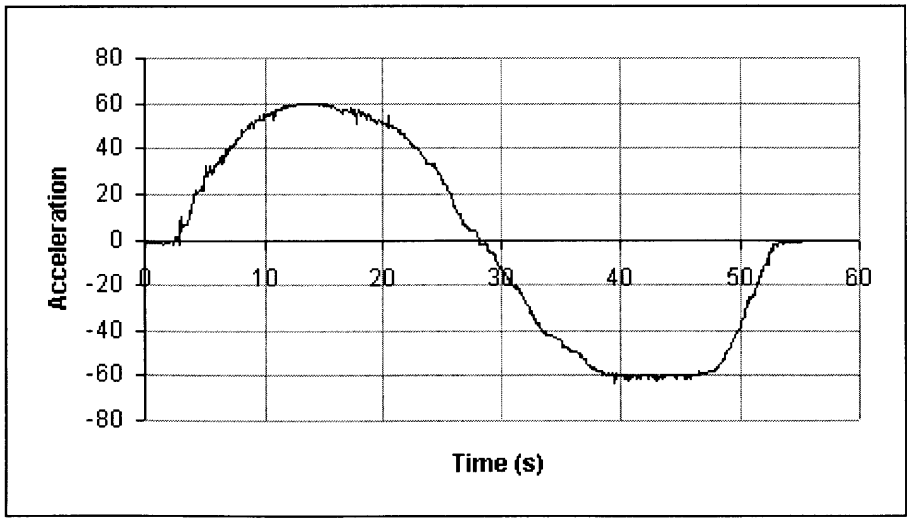


Figure 4.5: Calibration data for MICA mote

4.3.2.2 - Frequency Sweep

The second set of three excitations involves a frequency sweep from low to high. This results in an exponential increase in acceleration, which is proportional to frequency squared.

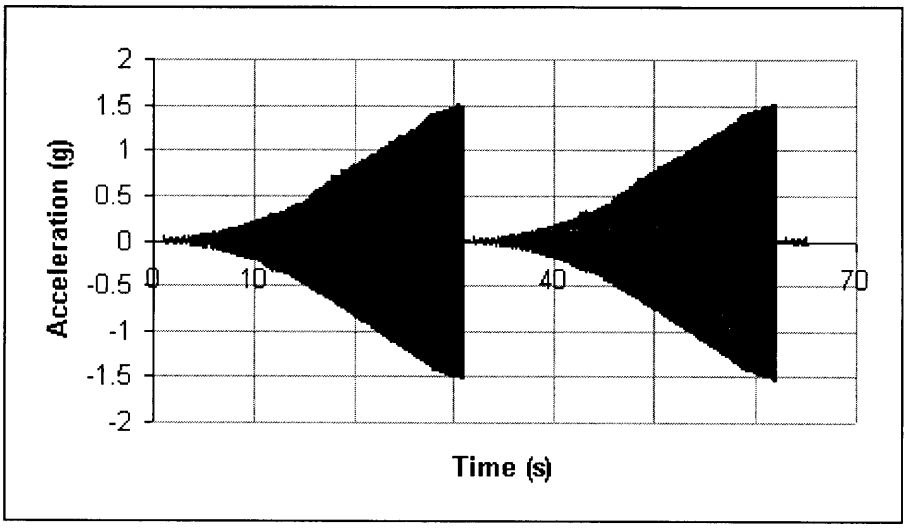


Figure 4.6: Frequency sweep input to shake table accelerometer

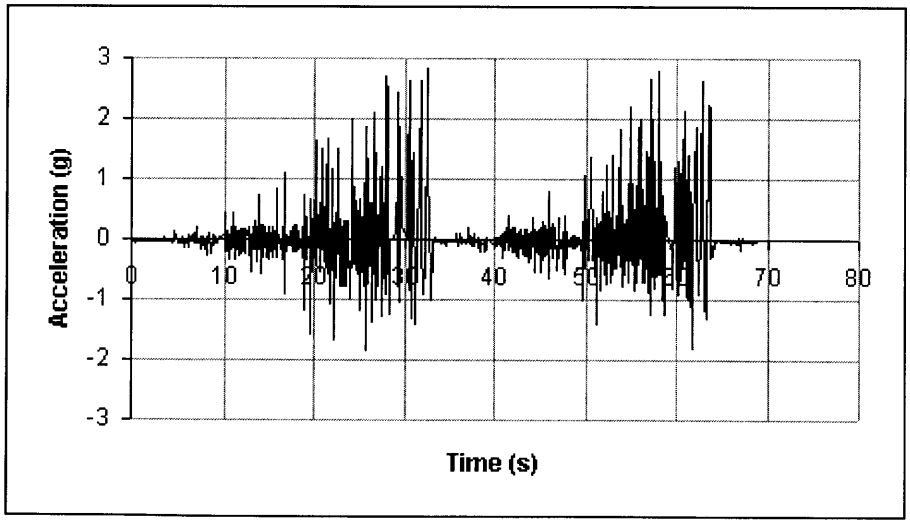


Figure 4.7: Result of frequency sweep #1

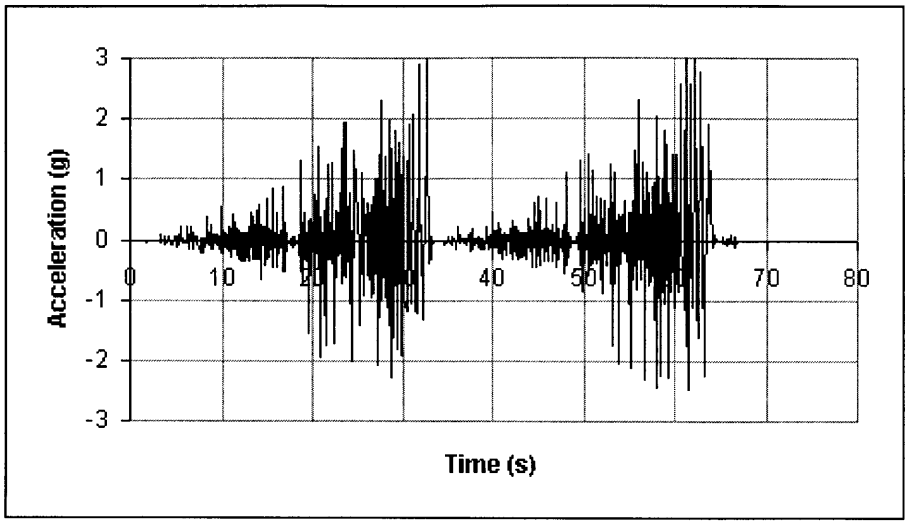


Figure 4.8: Result of frequency sweep #2

Figure 4.6 shows the acceleration retrieved from the accelerometer built into the shaking table. This clearly demonstrates the frequency sweep that is being applied. Figures 4.7 and 4.8 show the output from the sensor network for two different runs of the frequency sweep loading. There is a notable difference between the results from the sensors and the excitation, and between the results of the two sensors.

4.3.2.3 - Sample Earthquake

The third set of three excitations involves subjecting the mote to a sample earthquake. In this case, the excitation is a damped version of the Kobe earthquake.

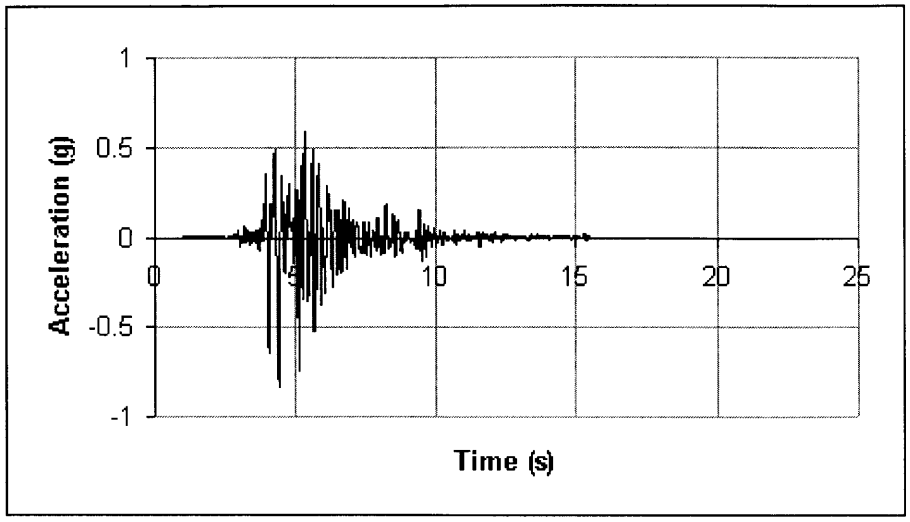


Figure 4.9: Kobe damped input to shake table accelerometer

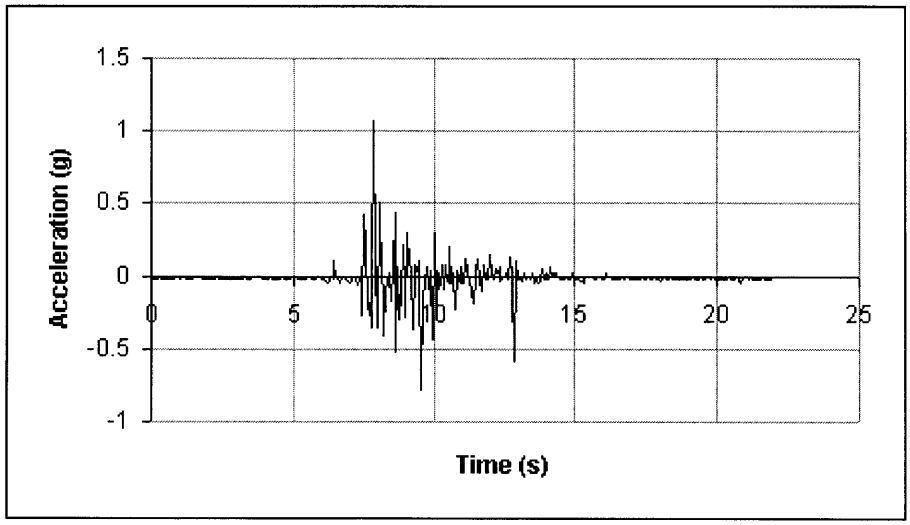


Figure 4.10: Result of Kobe damped earthquake #1

As shown in Figures 4.9 and 4.10, there are some differences between the input and output waveforms for an earthquake loading, however the differences are not nearly as obvious as with the frequency sweep, and are somewhat alleviated when it is realized that the signs of the two different waveforms are opposite. This was a mistake that will be considered in future tests. It is important to note this is only a major issue for non-symmetric loading, such as from an earthquake.

4.3.2.4 - Sine Waves

The fourth set of excitations involves a series of three sine wave excitations with different frequencies and amplitudes.

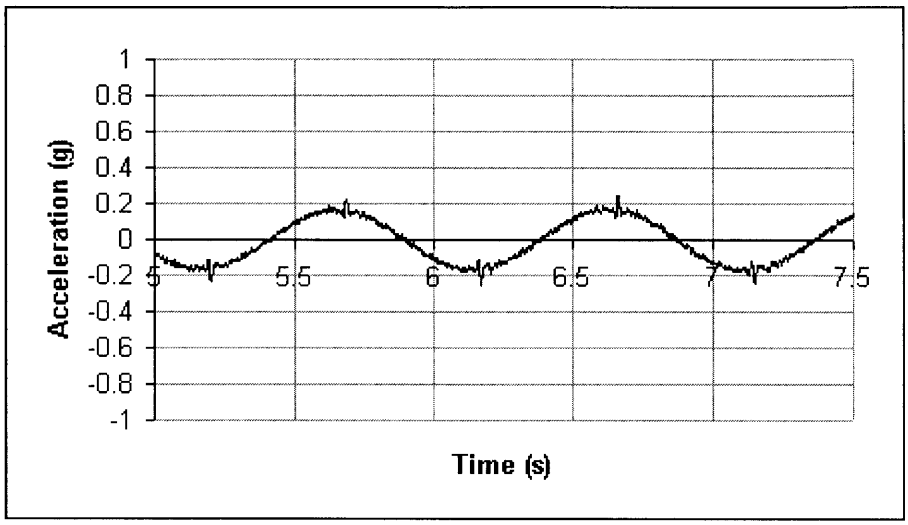


Figure 4.11: Sine loading #1 input to shake table accelerometer

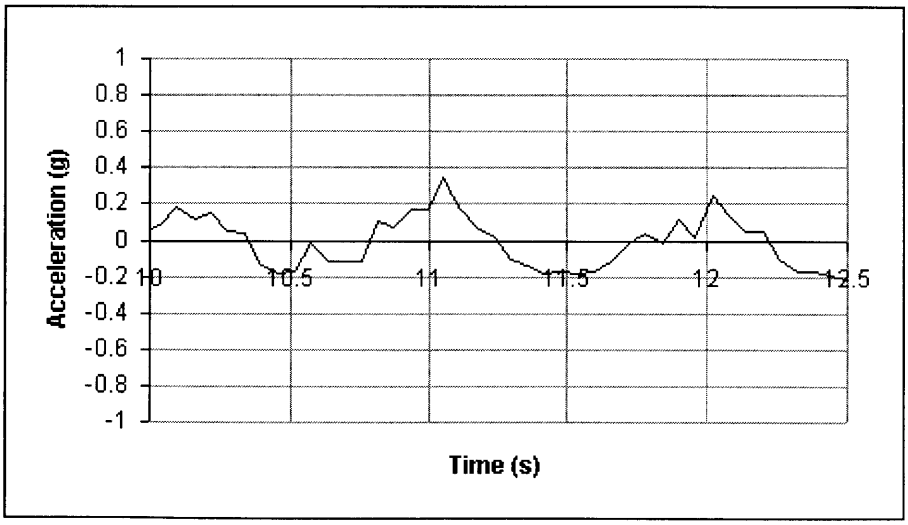


Figure 4.12: Result of sine loading #1

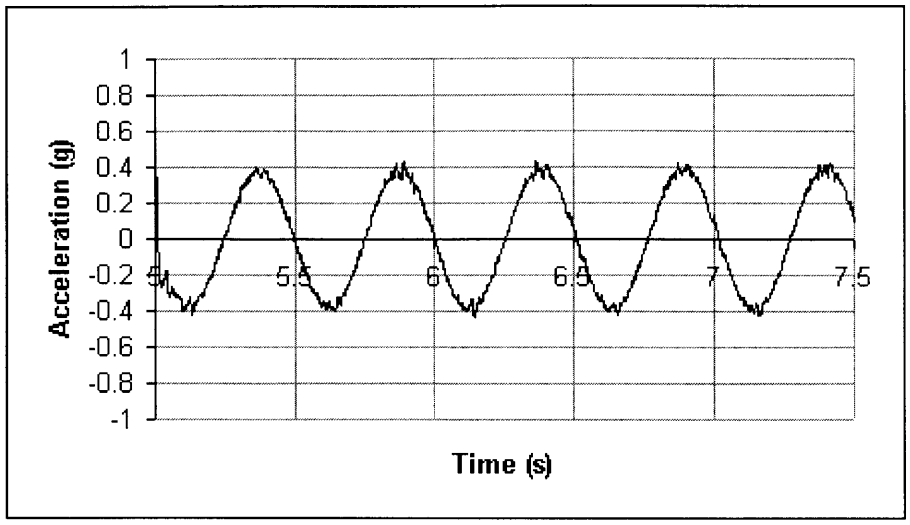


Figure 4.13: Sine loading #2 input to shake table accelerometer

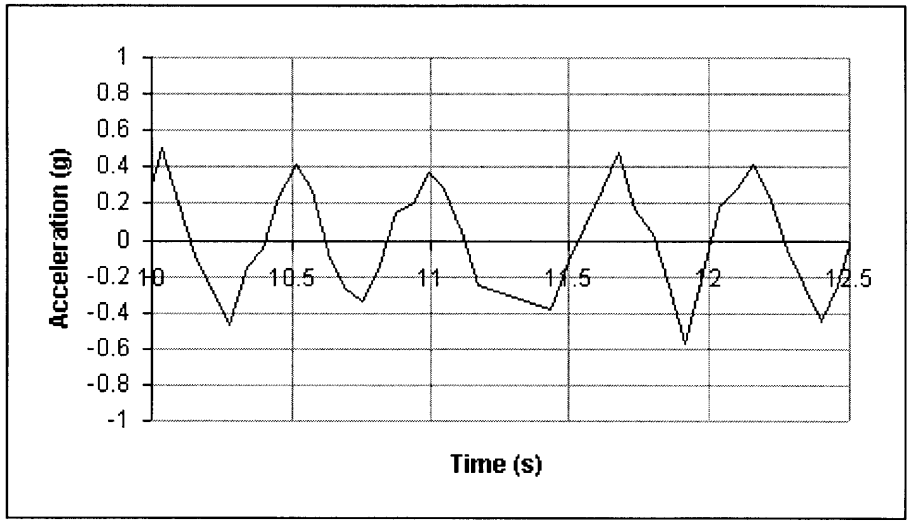


Figure 4.14: Result of sine loading #2

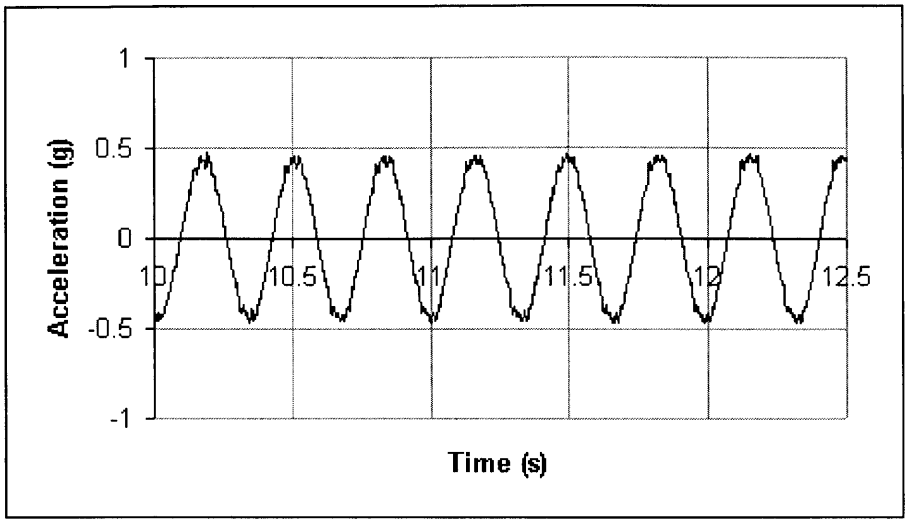


Figure 4.15: Sine loading #3 input to shake table accelerometer

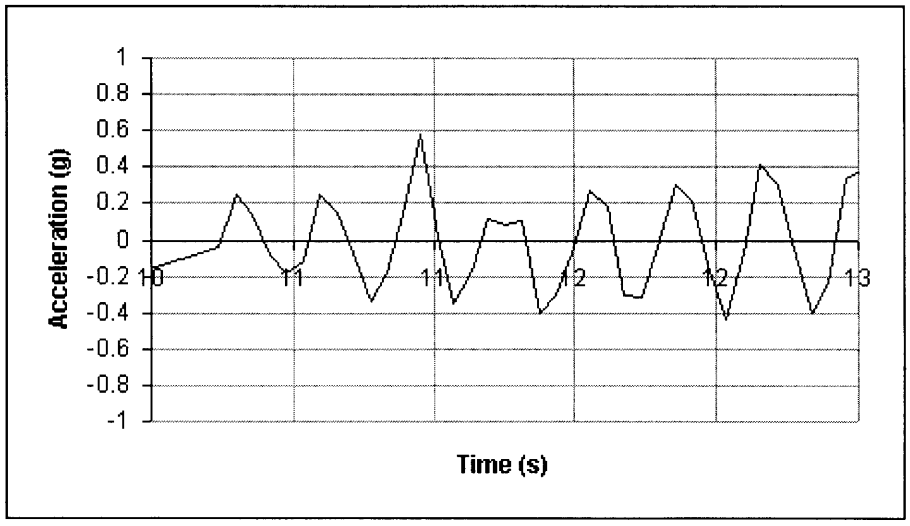


Figure 4.16: Result of sine loading #3

The input and output waveforms for sinusoidal loading are shown in Figure 4.11 and Figure 4.12, Figure 4.13 and Figure 4.14, and Figure 4.15 and Figure 4.16. There are some obvious differences between the input and output in general. However, it should be noted that the output waveform is relatively consistent in capturing the correct frequency and amplitude.

4.3.3 - Results

Overall, the results from the experiments are less than perfect. From a purely visual inspection, it is obvious that the outputs from the sensor network are significantly different from the input excitations. General waveform shapes are similar, but there are gaps and variations. There are three main possible causes for these discrepancies.

The sampling frequency of the sensing device is of primary concern. Based on the output data, the sensing mote samples data every 0.06 seconds. This is equivalent to a frequency of 16.7 Hz. The sampling frequency creates a natural limit to frequencies that can be measured. This limit is exceeded by a large amount in the frequency sweep, and most likely in the Kobe damped earthquake loading. Generally, a sampling frequency that is too low results in missed data.

The wireless link can also be a source of lost data. The data packet protocol has no method of ensuring that data is received. Any lost packets result in the contained data being lost permanently. These lost packets appear as long straight lines in the output. Furthermore, the wireless link creates a limit to the possible sampling frequency, as it is pointless to sample more often if the data cannot be sent. This exact limit is unknown, but would be an interesting issue to investigate.

The final possible cause in discrepancy is the limit to sensitivity created by the analog to digital converter. The 10-bit ADC has natural limit as to the precision of the data it can record. This precision can in general be applied over a variety of ranges, but the range for the MICA mote has been hardwired and cannot be adjusted. The ADC has been designed to function with all of the sensing elements on the sensing node, and, as such, functions adequately for all of them, but not perfectly for any. For the accelerometer, the ADC offers a

precision of 1/60 g or 0.0167g. This limit in precision combined with the limited sampling rate combine to create a jagged appearance to the outputs as seen most noticeably in Figure 4.12.

Chapter 5 - Conclusions

5.1 - Overview

From the work that has been done, we can discuss the applicability of wireless nodal sensing for civil engineering. This will be done with specific emphasis on the use of MICA motes, but include what has been learned and how this reflects on the use of wireless nodal sensing in general.

5.2 - Assessment of MICA motes

Based on the results obtained from the experimental work, it would seem that MICA motes could have some application for physical infrastructure, but this is tempered by a large number of restrictions. We will look at the good and bad points associated with the MICA motes.

One of the good points associated with the MICA motes is the relative simplicity involved in setting up a network. A user who understands computer programming can understand the related code and initiate primary functionality without too much difficulty. This is mostly related to the fact that both TinyOS and nesC are very well designed. They work reliably and provide easy access for understanding and manipulation.

Also, the wireless network was relatively consistent. Once a connection was established, it was quite consistent with only occasional lost packets.

However, there are two major factors that work against the MICA motes. The first has been mentioned, and is the sensitivity of the accelerometer. The sensitivity is limited because the analog-to-digital converter and its functionality

are hard-wired into the system. This means that it is not possible to tailor that aspect of the operation to the specific application. Generally, the sensitivity of the accelerometer is sufficient for earthquake- or impulse-type excitation, but is inadequate for measuring lower amplitudes. Microtremors and ambient vibrations, which might be present on a construction site, have amplitudes that are too small to be read and distinguished from the noise of the system. This is a limiting factor in the motes applicability for a great number of physical infrastructure applications.

Also, the motes suffer from a lack of robustness. While the sensor worked well in a controlled lab setting, there is concern about its ability to withstand real environmental conditions. Providing a casing could be a simple solution, though it is expected that this would interact poorly with the wireless data transmission.

Furthermore, the MICA motes have not been designed to deal with large accelerations, which could be present in physical infrastructure systems. This is most obvious in the connection between the processing board and the power supply. This connection consists of two leads and some two-sided tape. While this structure provides adequate shear connection, it provides very little torsional resistance. This makes the connections vulnerable, makes the mote difficult to mount, and creates doubt as to whether the sensor is seeing the same loading as the object on which it is mounted.

There is also some question as to the structural ability of the 51-pin connector between the processing board and the sensing board. It was sometimes found, during aggressive vibrations, that the wireless connection between the sensing mote and the base station would be lost. Generally, the application of pressure to the connector would remedy the situation. However, if this connector came loose in the field, as it might due to motion or temperature, it could cause a significant loss of data.

The result of this work indicates that MICA motes are not the appropriate sensing mechanism for physical infrastructure. They lack the reliability and sensitivity required for most applications. However, they have shown some interesting functionality, which shows potential for network embedded sensors as a whole. Though MICA motes are not the appropriate technology, it is believed that wireless nodal sensing will form an important part in infrastructure monitoring, provided technology is developed appropriately.

This leads to the question that if MICA motes are not the appropriate technology, what are we looking for?

5.3 - Future Technology

The major problem with the motes is their generic nature. They have been designed to supply a number of different sensing mechanisms, and, as such, are not well designed for any specific application.

One of the requirements listed for a sensing system in Chapter 2 was flexibility. It is important that a sensing system can be used for multiple applications in order to make its development economically feasible. However, the method of flexibility used on the MICA motes is not appropriate.

The motes offer flexibility by providing a number of different sensing mechanisms, which can be varied to a limited degree, by changing the sensing board. However, this modularity has shown itself to be too coarse. It would be useful to be able to change components, such as the ADC, in order to customize performance for a specific application. So, while flexibility is important, it should be provided by allowing system configuration, not by attempting to provide more functionality than is required.

One of the main assets of the motes is the TnyOS and nesC applications. They offer a robust application development structure. If these could be maintained, it would be an asset, though not a priority.

In general, an appropriate technology would consist of a similar system with more control over the hardware components. This is not currently an option. However, there are some technologies that are going this way, and may provide possibilities.

5.3.1 - Next generation of UC Berkeley motes

The mote family has continued to grow beyond the MICA motes that were used in this work. MICA2 and MICA2dot are now available for purchase from Crossbow Technologies, Inc. However, these do not offer significant functionality beyond that of the MICA mote. Academic work in the area has been focused on reducing the size of the radio and processor module for the mote. This is useful advancement and will ultimately provide greater functionality and decreased cost. However, it is not expected that the sensing module will become more modular.

The mote line has been developed for research purpose, and functions well as such. However, the motes are not intended for commercial use, and will most likely never be completely applicable.

5.3.2 – Millennial Net iBean

The iBean from Millenial Net offers a great deal of potential for network embedded sensors. The iBeans can act as radio transmitters and receivers for a network. They have a proprietary application, which provides robust wireless

data transmission, hopping, and reception. The iBean network is self-configuring, and will use the best radio path to send its data to the base station. This intelligence also provides the ability to detect if a single bean's radio has stopped functioning.

However, there are some disadvantages to working with the iBeans. While they are robust, the proprietary nature of the code means that it cannot be adjusted. Furthermore, there is no established link to connect sensors to the motes. This connection must be built and has to work around the capabilities of the beans. It also cannot use any established systems such as TinyOS.

The iBeans offer a good compromise between prepackaged and reconfigurable systems. Recognizing this, a group in the Civil and Environmental Engineering Department at MIT has been working to develop this technology such that it can be used to monitor decaying tunnels in the London Underground.

5.4 - Conclusions

Wireless nodal sensing offers a lot of potential for use in civil engineering. It is vitally important that civil engineers keep abreast of the technology in this rapidly changing field, or the application of the devices for a field as unique as physical infrastructure will be overlooked.

Sensing and monitoring in general, and in specific as they relate to physical infrastructure, offer a lot of opportunity for growth and advancement. Civil engineering, often viewed as a shrinking profession, should take this opportunity to grab this field and make it its own.

References

Burrer, C., Esteve, J., and Lora-Tamayo, E. Resonant Silicon Accelerometers in Bulk Micromachining Technology - An Approach. Journal of Microelectromechanical Systems. Vol. 5, No. 2, June 1996, pp 122-130.

Cheekiralla, S., Lakshmanan, H., Chen, Y.-J., Varadharajan, C., and Choudhary, V.S. Wireless Sensor-based Infrastructure Monitoring. 1.961 Project Report. Department of Civil and Environmental Engineering. 2002.

Chopra, A.K. Dynamics of Structures: Theory and Applications to Earthquake Engineering. Prentice-Hall, Inc. Upper Sandle River, New Jersey. 1995.

DeVoe, D.L. and Pisano, A.P. Surface Micromachined Piezoelectric Accelerometers (PiXLs). Journal of Microelectromechanical Systems. Vol. 10, No. 2, June 2001, pp 180-186.

Gardner, J.W. Microsensors: Principles and Applications. Wiley and Co. Chichester, New York. 1994.

Liu, C.-H., Barzilai, A.M., Reynolds, J.K., Partridge, A., Kenny, T.W., Grade, J.D., and Rockstad, H.K. Characterization of a High-Sensitivity Micromachined Tunneling Accelerometer with Micro-g Resolution. Journal of Microelectromechanical Systems. Vol. 7, No. 2, June 1998, pp 235-244.

Madou, M. Fundamentals of Microfabrication. CRC Press LLC. Boca Raton, Florida. 1997.

Partridge, A., Reynolds, J.K., Chui, B.W., Chow, E.M., Fitzgerald, A.M., Zhang, L., Maluf, N.I., and Kenny, T.W. A High-Performance Planar Piezoresistive Accelerometer. Journal of Microelectromechanical Systems. Vol. 9, No. 1, Mar. 2000, pp 58-66.

Quanser Consulting Inc. UCIST Shake Table II Instruction Manual.

Selvakumar, A. and Najafi, K. A High-Sensitivity Z-Axis Capacitive Silicon Microaccelerometer with a Torsional Suspension. Journal of Microelectromechanical Systems. Vol. 7, No. 2, June 1998, pp 192-200.

Strang, G. and Nguyen, T. Wavelets and Filter Banks, second ed. Wellesley-Cambridge Press. Wellesley, Massachusetts. 1997.

Appendix A

Application Code

A.1 Application Code for Oscilloscope

File: Oscilloscope.nc

```
/*                                                                      tab:4
 *
 *
 * "Copyright (c) 2000-2002 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation for any purpose, without fee, and without written agreement is
 * hereby granted, provided that the above copyright notice, the following
 * two paragraphs and the author appear in all copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 */
/*                                                                      tab:4
 *  IMPORTANT: READ BEFORE DOWNLOADING, COPYING, INSTALLING OR USING.  By
 *  downloading, copying, installing or using the software you agree to
 *  this license.  If you do not agree to this license, do not download,
 *  install, copy or use the software.
 *
 *  Intel Open Source License
 *
 *  Copyright (c) 2002 Intel Corporation
 *  All rights reserved.
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions are
 *  met:
 *
 *        Redistributions of source code must retain the above copyright
 *  notice, this list of conditions and the following disclaimer.
 *        Redistributions in binary form must reproduce the above copyright
 *  notice, this list of conditions and the following disclaimer in the
 *  documentation and/or other materials provided with the distribution.
 *      Neither the name of the Intel Corporation nor the names of its
 *  contributors may be used to endorse or promote products derived from
 *  this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 *  ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
```

```
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 * PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE INTEL OR ITS
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *
 */

/*
  $Id: Oscilloscope.nc,v 1.2 2002/12/07 20:00:49 yrchen Exp $
  $Log: Oscilloscope.nc,v $
  Revision 1.2  2002/12/07 20:00:49  yrchen
  works with accel.

  Revision 1.1.1.3  2002/12/06 19:51:53  yrchen
  Working 3-level decomposition of cubic B-spline filter bank.

*/

includes OscopeMsg;

/* Comment out the following if you want to use Photo instead of Accel */
#define USE_ACCEL


configuration Oscilloscope { }
implementation
{
  components Main, OscilloscopeM, TimerC, //ClockC,
    LedsC, DWT1C, DWT2C, DWT3C,
#ifdef USE_ACCEL
    Accel,
#else
    Photo as Data,
#endif
    GenericComm as Comm;

  Main.StdControl -> OscilloscopeM;
  OscilloscopeM.DWT1 -> DWT1C;
  OscilloscopeM.DWT2 -> DWT2C;
  OscilloscopeM.DWT3 -> DWT3C;
  //OscilloscopeM.Clock -> ClockC;
  OscilloscopeM.Timer -> TimerC.Timer[unique("Timer")];
  OscilloscopeM.Leds -> LedsC;
#ifdef USE_ACCEL
  OscilloscopeM.SensorControl -> Accel;
  OscilloscopeM.ADC -> Accel.AccelX;
#else
```

```
  OscilloscopeM.SensorControl -> Data;
  OscilloscopeM.ADC -> Data;
#endif
  OscilloscopeM.CommControl -> Comm;
  OscilloscopeM.ResetCounterMsg -> Comm.ReceiveMsg[AM_OSCOPERESETMSG];
  OscilloscopeM.DataMsg -> Comm.SendMsg[AM_OSCOPEMSG];
}
```

File OscilloscopeM.nc

```
/*                                                                      tab:4
 *
 *
 * "Copyright (c) 2000-2002 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation for any purpose, without fee, and without written agreement is
 * hereby granted, provided that the above copyright notice, the following
 * two paragraphs and the author appear in all copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 */
/*                                                                      tab:4
 *  IMPORTANT: READ BEFORE DOWNLOADING, COPYING, INSTALLING OR USING.  By
 *  downloading, copying, installing or using the software you agree to
 *  this license.  If you do not agree to this license, do not download,
 *  install, copy or use the software.
 *
 *  Intel Open Source License
 *
 *  Copyright (c) 2002 Intel Corporation
 *  All rights reserved.
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions are
 *  met:
 *
 *        Redistributions of source code must retain the above copyright
 *  notice, this list of conditions and the following disclaimer.
 *        Redistributions in binary form must reproduce the above copyright
 *  notice, this list of conditions and the following disclaimer in the
 *  documentation and/or other materials provided with the distribution.
 *      Neither the name of the Intel Corporation nor the names of its
 *  contributors may be used to endorse or promote products derived from
 *  this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 *  ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 *  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 *  PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE INTEL OR ITS
 *  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
```

```
 *  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 *  PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 *  PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 *  LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 *  NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 *  SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *
 */
/*
 * Authors:   Jason Hill
 * History:   created 10/5/2001
 */
/*
  $Id: OscilloscopeM.nc,v 1.9 2003/02/14 16:44:01 yrchen Exp $
  $Log: OscilloscopeM.nc,v $
  Revision 1.9  2003/02/14 16:44:01  yrchen
  added #define DRIFT_CORRECTION

  Revision 1.8  2002/12/09 03:53:39  yrchen
  * tested with double and int32_t as data_t
  * no more overflows given the 10-bit ADC output
  * use of prescaling of input signal when data_t is int32_t
  * successfully removed baseline drift.
  * ignore some number of samples initially to get a better estimate of velocity
  * computed velocity still needs normalizing upon displaying

  Revision 1.7  2002/12/09 01:33:24  yrchen
  Added data_t defined in datatype.h.

  Revision 1.6  2002/12/08 18:30:19  yrchen
  still has overflow issue...

  Revision 1.5  2002/12/08 16:55:51  yrchen
  Distinguish between rounding and flooring ( incomplete ) in DWTC.h

  Revision 1.4  2002/12/08 06:59:10  yrchen
  Added (1/2,2) normalization factor to the end of a one-level Cubic B-spline
  decomposition, to avoid overflowing INT16_T after three levels of decomposition.

  Revision 1.3  2002/12/08 04:57:47  yrchen
  Removed ^M. Extracted dwt and idwt codes to DWTC.h.
  Use of #include "DWTC.h" in all DWT?C.nc.

  Revision 1.2  2002/12/07 20:00:49  yrchen
  works with accel.

  Revision 1.1.1.4  2002/12/06 19:51:53  yrchen
  Working 3-level decomposition of cubic B-spline filter bank.

*/

includes OscopeMsg;
```

```
/**
 * This module implements the OscilloscopeM component, which
 * periodically takes sensor readings and sends a group of readings
 * over the radio. BUFFER_SIZE defines the number of readings sent
 * in a single packet. The Yellow LED is toggled whenever a new
 * packet is sent, and the red LED is turned on when the sensor
 * reading is above some constant value.
 */
module OscilloscopeM
{
  provides interface StdControl;
  uses {
    interface DWT as DWT1;
    interface DWT as DWT2;
    interface DWT as DWT3;
    //interface Clock;
    interface Timer;
    interface Leds;
    interface StdControl as SensorControl;
    interface ADC;
    interface StdControl as CommControl;
    interface SendMsg as DataMsg;
    interface ReceiveMsg as ResetCounterMsg;
  }
}
#include "datatype.h"
//#define _SOFT_THRESH_
#define INITIAL_DELAY 100  /* the initial number of samples to ignore after
                              DWT filtering */
// Comment out SEND_VELOCITY if you want accel.
#define SEND_VELOCITY

// Comment out DRIFT_CORRECTION if you want uncorrected accel or velocity.
#define DRIFT_CORRECTION

//=========================================================
==========
#ifdef DRIFT_CORRECTION
#  define __PRE_SCALING__
#  define PRESCALE_FACTOR 12

#else
#  warning
#  warning "DRIFT CORRECTION DISABLED!"
#  warning "DRIFT CORRECTION DISABLED!"
#  warning "DRIFT CORRECTION DISABLED!"
#  warning
#endif
implementation
{
  int16_t g_time_n;
  data_t velocity;
```

```
  data_t outvar;
  uint16_t readingNumber;
  TOS_Msg g_msg[2];
  uint8_t g_currentMsg;

  /* ================================================== */
  // declare module static variables here
#define BUF_SIZE 64
#define MASKDATA 0x3f    /* should be (BUF_SIZE-1) */

  data_t g_rdata0[BUF_SIZE],
    g_rdata1[BUF_SIZE],
    g_rdata11[BUF_SIZE],
    g_rdata2[BUF_SIZE],
    g_rdata21[BUF_SIZE];  // circular buffer

  data_t g_ahat0[2], g_ahat1[2], g_ahat2[2];

  // index to the head of the circular buffer
  int16_t g_head0, g_head1, g_head11, g_head2, g_head21;

  inline void putdata(data_t val) {
    int16_t p = g_head0;

    // 'head' is where the data can be READ.
    // Need to advance by one before WRITE.
    g_head0 = (p+1) & MASKDATA;
    g_rdata0[g_head0] = val;

  }

  inline void putdata1(data_t val) {
    int16_t p = g_head1;

    // 'head' is where the data can be READ.
    // Need to advance by one before WRITE.
    g_head1 = (p+1) & MASKDATA;
    g_rdata1[g_head1] = val;

  }

  inline void putdata11(data_t val) {
    int16_t p = g_head11;

    // 'head' is where the data can be READ.
    // Need to advance by one before WRITE.
    g_head11 = (p+1) & MASKDATA;
    g_rdata11[g_head11] = val;

  }

  inline void putdata2(data_t val) {
    int16_t p = g_head2;
```

```
  // 'head' is where the data can be READ.
  // Need to advance by one before WRITE.
  g_head2 = (p+1) & MASKDATA;
  g_rdata2[g_head2] = val;

}

inline void putdata21(data_t val) {
  int16_t p = g_head21;

  // 'head' is where the data can be READ.
  // Need to advance by one before WRITE.
  g_head21 = (p+1) & MASKDATA;
  g_rdata21[g_head21] = val;

}

void data_to_packet(TOS_Msg msg[], //struct OscopeMsg *pack,
                    uint8_t *currentMsg,
                    uint16_t *readingNbr,
                    uint16_t channel,
                    int16_t data) // was: UINT16_T data
  {
    struct OscopeMsg *pack = (struct OscopeMsg *)msg[*currentMsg].data;
    /* ====================================== */
    //      pack->data[packetReadingNbr++] = data;
    pack->data[(*readingNbr)%BUFFER_SIZE] = data;
    (*readingNbr)++;

    /* -------------------------------------- */
    // THE SIGN PROBLEM was FIXED by modifying //
    // tos/lib/OscopeMsg.h                    //
    /* -------------------------------------- */


    /* If we have filled in enough readings... */
    //      if (packetReadingNumber == BUFFER_SIZE)
    if ( ((*readingNbr) % BUFFER_SIZE == 0) & (*readingNbr)>0 )
        {
          //        packetReadingNbr = 0;
          pack->channel = channel;
          pack->lastSampleNumber = *readingNbr;
          pack->sourceMoteID = TOS_LOCAL_ADDRESS;


          /* Try to send the packet. Note that this will return
           * failure immediately if the packet could not be queued for
           * transmission.
           */


          // TURN ON the radio
```

```
        //call CommControl.start();

        if (call DataMsg.send(TOS_BCAST_ADDR, sizeof(struct OscopeMsg),
                              &msg[*currentMsg]))
          {
            (*currentMsg) ^= 0x1;
            call Leds.yellowToggle();
          }

        // TURN OFF the radio
        //call CommControl.stop();
      }
  }

 /* ================================================== */
 void filter_data() {
   data_t a[2], a1[2], a2[2];
   data_t out0[2], out1[2], out2[2];
   //    int16_t ahat1[2];

   if ((g_time_n & 0x1)==0) {
     //effectively downsample by 2
     // S/P converter
     a[0] = g_rdata0[g_head0];
     a[1] = g_rdata0[(g_head0-1) & MASKDATA];

     call DWT1.dwt(a,out0);

#if 1
     putdata1 (out0[0]); // into g_rdata1[]
     //putdata1 (g_time_n%15); // into g_rdata1[]
     putdata11(out0[1]); // into g_rdata11[]

     if ((g_time_n & 0x3) == 0) {
         // S/P converter
         a1[0] = g_rdata1[g_head1];
         a1[1] = g_rdata1[(g_head1-1) & MASKDATA];

         call DWT2.dwt(a1, out1);

         // ============= BEGIN OF LEVEL-3 =============
         // ============= BEGIN OF LEVEL-3 =============
         putdata2 (out1[0]);
         putdata21(out1[1]);

         if ((g_time_n & 0x7) == 0) {
           a2[0] = g_rdata2[g_head2];
           a2[1] = g_rdata2[(g_head2-1) & MASKDATA];

           call DWT3.dwt(a2, out2);
           dbg(DBG_USR2, "DWT_LPF=%d, HPF=%d\n", out2[0],out2[1]);
           out2[0]=0;  // to null low frequencies
           call DWT3.idwt(out2, g_ahat2);
```

```
        }

        out1[0] = g_ahat2[((g_time_n & 0x7)==0) ? 1 : 0];
        out1[1] = g_rdata21[(g_head21-7)&MASKDATA];
        // ============= END OF LEVEL-3 ==============
        // ============= END OF LEVEL-3 ==============

        call DWT2.idwt(out1, g_ahat1);
    }

#   if 0
    dbg(DBG_USR1, "%d, data=%d, ahat=%d\n",
          g_time_n, g_rdata1[g_head1], g_ahat1[((g_time_n & 0x3)==0)?1:0]);
#   endif
    out0[0] = g_ahat1[((g_time_n & 0x3)==0) ? 1 : 0];
    out0[1] = g_rdata11[(g_head11-21)&MASKDATA];

#endif

    call DWT1.idwt(out0,g_ahat0);

#if 0
    data_to_packet(g_msg, &g_currentMsg, &readingNumber, 2, g_ahat0[1]);
    data_to_packet(g_msg, &g_currentMsg, &readingNumber, 2, g_ahat0[0]);
#endif
  } // endif ((g_time_n & 0x1)==0)

#if 1
  /*
    This verifies that the delay is 7 samples [(M-1)+M*\ell]
    with M=2 and \ell=3 for Cubic B-spline
  */
  // This one shows the delay of 7 samples
  // if only one level of decomposition
  dbg(DBG_USR1, "%d, data=%lf, ahat=%lf\n",
        g_time_n, g_rdata0[g_head0], g_ahat0[((g_time_n & 0x1)==0)?1:0]);

  // This one shows how to delay the input signal to compare ...
  //dbg(DBG_USR1, "%d, data[n-7]=%d, ahat[n]=%d\n",
  //g_time_n, g_rdata0[(g_head0-7)&MASKDATA],
  //g_ahat0[((g_time_n & 0x1)==0)?1:0]);
#endif
 }



 /*
==========================================================
*/


 /**
  * Used to initialize this component.
```

```
 */
command result_t StdControl.init() {
  int i;
  call DWT1.init();
  call DWT2.init();
  call DWT3.init();
  call Leds.init();
  call Leds.yellowOff(); call Leds.redOff(); call Leds.greenOff();

  //turn on the sensors so that they can be read.
  call SensorControl.init();

  call CommControl.init();

  //call Clock.setRate(TOS_I8PS, TOS_S8PS);
  /* -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= */
  g_time_n = 0;
  velocity = 0;

  for (i=0; i<BUF_SIZE; i++) {
    g_rdata0[i]=0;
    g_rdata1[i]=0;
    g_rdata11[i]=0;
    g_rdata2[i]=0;
    g_rdata21[i]=0;
  }

  //    for(i=0;i<8;i++) {
  //    }

  g_head0 =
    g_head1 = g_head11 =
    g_head2 = g_head21 =
    -1;

  /* -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= */

  g_currentMsg = 0;
  //    packetReadingNumber = 0;
  readingNumber = 0;
  //    readingNumber1 = 0;

  dbg(DBG_BOOT, "OSCOPE initialized\n");
  return SUCCESS;
}

/**
 * Starts the SensorControl and CommControl components.
 * @return Always returns SUCCESS.
 */
command result_t StdControl.start() {
  call SensorControl.start();
```

```
    call CommControl.start();
    call Timer.start(TIMER_REPEAT, 50);
    return SUCCESS;
  }

  /**
   * Stops the SensorControl and CommControl componets.
   * @return Always returns SUCCESS.
   */
  command result_t StdControl.stop() {
    call SensorControl.stop();

    call CommControl.stop();
    call Timer.stop();
    return SUCCESS;
  }


  /**
   * Signalled when data is ready from the ADC. Stuffs the sensor
   * reading into the current packet, and sends off the packet when
   * BUFFER_SIZE readings have been taken.
   * @return Always returns SUCCESS.
   */
  event result_t ADC.dataReady(uint16_t data) {
    data_t sdata;
    data_t accel;


    //sdata = (g_time_n & 0x1f);

    sdata = //10;
      (signed)data;// - 538;

#ifdef DRIFT_CORRECTION
#  ifdef __PRE_SCALING__
    sdata <<= PRESCALE_FACTOR;  // PRE-SCALING
#  endif
    putdata(sdata);  // into the circular buffer

    filter_data();

    //data_to_packet(g_msg, &g_currentMsg, &readingNumber, 2, sdata);
    //data_to_packet(g_msg, &g_currentMsg, &readingNumber, 2, sdata-538);
    //data_to_packet(g_msg, &g_currentMsg, &readingNumber, 2, -5);

    accel = g_ahat0[((g_time_n & 0x1)==0)?1:0];
    //accel = (data-538);
#else
    accel = sdata-538; // This is unscaled.
#endif  /* DRIFT_CORRECTION */

#ifdef _SOFT_THRESH_
```

```
    // CAN IMPLEMENT SOFT THRESHOLD ON sdata TO REMOVE
    // SMALL FLUCTUATIONS ===> This causes drifting of velocity!!!
    // Soft thresholding...
    if (accel<(-12)) accel += 12;
    else if (accel>12) accel -=12;
    else accel=0;
#endif

    if (g_time_n<INITIAL_DELAY) {
      velocity = 0;
    } else {
      velocity += accel;
    }

#ifdef SEND_VELOCITY
    outvar = velocity;
#else
    outvar = accel;
#endif

#ifdef __PRE_SCALING__
    outvar /= (1<<PRESCALE_FACTOR);
#endif

    /* Can apply whatever normalization factor for velocity below */
    data_to_packet(g_msg, &g_currentMsg, &readingNumber, 1, outvar);
/*                      //accel */
/* #ifdef __PRE_SCALING__ */
/* #  ifdef SEND_VELOCITY */
/*                      velocity/(1<<PRESCALE_FACTOR) */
/* #else */
/*                      velocity */
/* #endif */
/*                      ); */

    g_time_n++;

    if (data > 0x0300)
      call Leds.redOn();
    else
      call Leds.redOff();

    return SUCCESS;
  }


  /**
   * Signalled when the previous packet has been sent.
   * @return Always returns SUCCESS.
   */
  event result_t DataMsg.sendDone(TOS_MsgPtr sent, result_t success) {
    return SUCCESS;
  }
```

```
  /**
   * Signalled when the clock ticks.
   * @return The result of calling ADC.getData().
   */
  event result_t Timer.fired() {
    return call ADC.getData();
  }

  /**
   * Signalled when the reset message counter AM is received.
   * @return The free TOS_MsgPtr.
   */
  event TOS_MsgPtr ResetCounterMsg.receive(TOS_MsgPtr m) {
    readingNumber = 0;
    return m;
  }
}
```

File: DWT.nc

```
/*
  $Id: DWT.nc,v 1.3 2002/12/09 01:33:24 yrchen Exp $
  $Log: DWT.nc,v $
  Revision 1.3  2002/12/09 01:33:24  yrchen
  Added data_t defined in datatype.h.

  Revision 1.2  2002/12/08 04:57:47  yrchen
  Removed ^M. Extracted dwt and idwt codes to DWTC.h.
  Use of #include "DWTC.h" in all DWT?C.nc.

  Revision 1.1.1.2  2002/12/06 19:51:53  yrchen
  Working 3-level decomposition of cubic B-spline filter bank.

*/
#include "datatype.h"
interface DWT {
  command result_t init();

  command result_t dwt(data_t a[],    // input polyphases
                       data_t out[]   // output polyphases
                       );

  command result_t idwt(data_t out[],  /* this is actually the input */
                        data_t a_hat[] /* reconstructed polyphases */
                        );

  command result_t test();
}
```

File: DWT1c.nc

(Note: Files DWT2c.nc and DWT3c.nc are the same as below except for the line "module DWT1C {" in which the appropriate file name should be substituted.)

```
/*
  Cubic B-spline Filter Bank
*/

/*
  $Id: DWT1C.nc,v 1.2 2002/12/08 04:57:47 yrchen Exp $
  $Log: DWT1C.nc,v $
  Revision 1.2  2002/12/08 04:57:47  yrchen
  Removed ^M. Extracted dwt and idwt codes to DWTC.h.
  Use of #include "DWTC.h" in all DWT?C.nc.

  Revision 1.1.1.2  2002/12/06 19:51:53  yrchen
  Working 3-level decomposition of cubic B-spline filter bank.

*/

module DWT1C {
  provides interface DWT;
}
#include "DWTC.h"
```

File: DWTC.h

```
/*
  $Log: DWTC.h,v $
  Revision 1.5  2002/12/09 01:33:24  yrchen
  Added data_t defined in datatype.h.

  Revision 1.4  2002/12/08 18:30:19  yrchen
  still has overflow issue...

  Revision 1.3  2002/12/08 16:55:51  yrchen
  Distinguish between rounding and flooring ( incomplete ) in DWTC.h

  Revision 1.2  2002/12/08 06:59:10  yrchen
  Added (1/2,2) normalization factor to the end of a one-level Cubic B-spline
  decomposition, to avoid overflowing INT16_T after three levels of decomposition.

  Revision 1.1  2002/12/08 04:57:47  yrchen
  Removed ^M. Extracted dwt and idwt codes to DWTC.h.
  Use of #include "DWTC.h" in all DWT?C.nc.

 */

// TODO: (1) rounding for negative numbers
//       (2) PR issue: don't divide by 2 in forward transform....
//           or consider using INT32_T for internal nodes and
//           getting rid of the division by 2 in forward transform.


//#define ROUND               /* DO NOT USE ROUND */
//#define __DWT_NORMALIZED__  /* Its use prevents PR */

#include "datatype.h"
implementation
{
  int16_t dummy;
  data_t z0[3], z1[3];

  command result_t DWT.init() {
    int i;

    dummy=0;

    // initially at rest
    for (i=0;i<3;i++) { z0[i]=z1[i]=0; }

    return SUCCESS;
  }

  /*
    Forward DWT
  */
  command result_t DWT.dwt(data_t a[],    // input polyphases
```

```
                                data_t out[]   // output polyphases
                                ) {
  data_t t0,t1;

#ifdef ROUND
  //    t0 = a[1] + ((a[0]+2)>>2) + ((z0[0]+2)>>2);
  t0 = a[1] + ((a[0]+2)/4) + ((z0[0]+2)/4);
#else
  //    t0 = a[1] + (a[0]>>2) + (z0[0]>>2);
  t0 = a[1] + (a[0]/4) + (z0[0]/4);
#endif
  t1 = (z0[0]) + (t0+(z0[1]));

#ifdef ROUND
  // Low-pass channel
  //    out[0] = (z0[2]+1) >> 1;
  out[0] = (z0[2]+1) /2;
  // High-pass channel
  //    out[1] = ( z0[1] - ( ((3*t1+8)>>4) + ((3*z0[2]+8)>>4)) ) << 1;
  out[1] = ( z0[1] - ( ((3*t1+8)/16) + ((3*z0[2]+8)/16)) ) *2L;
#else
  // Low-pass channel
  //out[0] = (z0[2]);// >> 1;
  out[0] = (z0[2]);// /2;
  // High-pass channel
  //out[1] = ( z0[1] - ( ((3*t1)>>4) + ((3*z0[2])>>4)) );// << 1;
  out[1] = ( z0[1] - ( ((3*t1)/16) + ((3*z0[2])/16)) );// *2L;
#endif

#ifdef __DWT_NORMALIZED__
  out[0] /= 2;
  out[1] *= 2;
#endif

  // update internal buffers
  z0[2] = t1;
  z0[1] = t0;
  z0[0] = a[0];

  return SUCCESS;
 }

 /*
  Inverse DWT
 */
 command result_t DWT.idwt(data_t out[],  /* this is actually the input */
                           data_t a_hat[] /* reconstructed polyphases */
                           ) {
  data_t t[3];
  data_t s[2];

#ifdef __DWT_NORMALIZED__
  out[0] *= 2;
```

```
    out[1] /= 2;
#endif

#ifdef ROUND
    //t[2] = out[1] + (s[0] = ((3*(out[0])+8) >> 4));
    t[2] = out[1] + (s[0] = ((3*(out[0])+8) /16));
#else
    //t[2] = out[1] + (s[0] = ((3*(out[0])) >> 4));
    t[2] = out[1] + (s[0] = ((3*(out[0])) /16));
#endif
    s[1] = s[0] + z1[2];

    //t[1] = (out[0]) - s[1];
    t[1] = (out[0]) - s[1];
    a_hat[0] = z1[1] - s[1];          // the even phase

#ifdef ROUND
    //t[0] = s[1] - ((a_hat[0]+2) >> 2);
    //a_hat[1] = z1[0] - ((a_hat[0]+2) >> 2);  // the odd phase
    t[0] = s[1] - ((a_hat[0]+2) /4);
    a_hat[1] = z1[0] - ((a_hat[0]+2) /4);  // the odd phase
#else
    //t[0] = s[1] - (a_hat[0] >> 2);
    //a_hat[1] = z1[0] - (a_hat[0] >> 2);  // the odd phase
    t[0] = s[1] - (a_hat[0] /4);
    a_hat[1] = z1[0] - (a_hat[0] /4);  // the odd phase
#endif

    // update internal buffers
    z1[2] = t[2];
    z1[1] = t[1];
    z1[0] = t[0];


    return SUCCESS;
  }

  command result_t DWT.test() {
    dummy++;
    dbg(DBG_USR1, "dummy=%d\n", dummy);

    return SUCCESS;
  }

}
```

A.2 Application Code for GenericBase

File: GenericBase.nc

```
/*                                                                    tab:4
 *
 *
 * "Copyright (c) 2000-2002 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation for any purpose, without fee, and without written agreement is
 * hereby granted, provided that the above copyright notice, the following
 * two paragraphs and the author appear in all copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 */
/*                                                                    tab:4
 *  IMPORTANT: READ BEFORE DOWNLOADING, COPYING, INSTALLING OR USING.  By
 *  downloading, copying, installing or using the software you agree to
 *  this license.  If you do not agree to this license, do not download,
 *  install, copy or use the software.
 *
 *  Intel Open Source License
 *
 *  Copyright (c) 2002 Intel Corporation
 *  All rights reserved.
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions are
 *  met:
 *
 *        Redistributions of source code must retain the above copyright
 *  notice, this list of conditions and the following disclaimer.
 *        Redistributions in binary form must reproduce the above copyright
 *  notice, this list of conditions and the following disclaimer in the
 *  documentation and/or other materials provided with the distribution.
 *      Neither the name of the Intel Corporation nor the names of its
 *  contributors may be used to endorse or promote products derived from
 *  this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 *  ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 *  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
```

```
 *  PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE INTEL OR ITS
 *  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 *  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 *  PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 *  PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 *  LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 *  NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 *  SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *
 */
configuration GenericBase {
}
implementation {
  components Main, GenericBaseM, RadioCRCPacket as Comm, UARTNoCRCPacket, LedsC;

  Main.StdControl -> GenericBaseM;

  GenericBaseM.UARTControl -> UARTNoCRCPacket;
  GenericBaseM.UARTSend -> UARTNoCRCPacket;
  GenericBaseM.UARTReceive -> UARTNoCRCPacket;

  GenericBaseM.RadioControl -> Comm;
  GenericBaseM.RadioSend -> Comm;
  GenericBaseM.RadioReceive -> Comm;

  GenericBaseM.Leds -> LedsC;
}
```

File: GenericBaseM.nc

```
/*                                                                      tab:4
 *
 *
 * "Copyright (c) 2000-2002 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation for any purpose, without fee, and without written agreement is
 * hereby granted, provided that the above copyright notice, the following
 * two paragraphs and the author appear in all copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR
 * DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT
 * OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF
 * CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 */
/*                                                                      tab:4
 *  IMPORTANT: READ BEFORE DOWNLOADING, COPYING, INSTALLING OR USING.  By
 *  downloading, copying, installing or using the software you agree to
 *  this license.  If you do not agree to this license, do not download,
 *  install, copy or use the software.
 *
 *  Intel Open Source License
 *
 *  Copyright (c) 2002 Intel Corporation
 *  All rights reserved.
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions are
 *  met:
 *
 *        Redistributions of source code must retain the above copyright
 *  notice, this list of conditions and the following disclaimer.
 *        Redistributions in binary form must reproduce the above copyright
 *  notice, this list of conditions and the following disclaimer in the
 *  documentation and/or other materials provided with the distribution.
 *      Neither the name of the Intel Corporation nor the names of its
 *  contributors may be used to endorse or promote products derived from
 *  this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 *  ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 *  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 *  PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE INTEL OR ITS
 *  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
```

```
 *  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 *  PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 *  PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 *  LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 *  NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 *  SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 *
 */
/* History:   created 1/25/2001
 *
 *
 */

/* Generic.Base.c
   - captures all the packets that it can hear and report it back to the UART
   - forward all incoming UART messages out to the radio
*/

module GenericBaseM {
  provides interface StdControl;
  uses {
    interface StdControl as UARTControl;
    interface BareSendMsg as UARTSend;
    interface ReceiveMsg as UARTReceive;

    interface StdControl as RadioControl;
    interface BareSendMsg as RadioSend;
    interface ReceiveMsg as RadioReceive;

    interface Leds;
  }
}
implementation
{
  TOS_Msg buffer;
  TOS_MsgPtr ourBuffer;
  bool sendPending;


  /* Generic.Base.Init:
     initialize lower components.
     initialize component state, including constant portion of msgs.
  */
  command result_t StdControl.init() {
    result_t ok1, ok2, ok3;

    ourBuffer = &buffer;
    sendPending = TRUE;

    ok1 = call UARTControl.init();
    ok2 = call RadioControl.init();
    ok3 = call Leds.init();
```

```
  sendPending = FALSE;

  dbg(DBG_BOOT, "GenericBase initialized\n");

  return rcombine3(ok1, ok2, ok3);
}

command result_t StdControl.start() {
  result_t ok1, ok2;

  ok1 = call UARTControl.start();
  ok2 = call RadioControl.start();

  return rcombine(ok1, ok2);
}

command result_t StdControl.stop() {
  result_t ok1, ok2;

  ok1 = call UARTControl.stop();
  ok2 = call RadioControl.stop();

  return rcombine(ok1, ok2);
}

TOS_MsgPtr receive(TOS_MsgPtr received, bool fromUART) {
  TOS_MsgPtr nextReceiveBuffer = received;

  dbg(DBG_USR1, "GenericBase received %s packet\n",
        fromUART ? "UART" : "radio");
  if ((!sendPending) &&
        (received->group == (TOS_AM_GROUP & 0xff))) {

    result_t ok;

    nextReceiveBuffer = ourBuffer;
    ourBuffer = received;
    dbg(DBG_USR1, "GenericBase forwarding packet to %s\n",
          fromUART ? "radio" : "UART");
    if (fromUART)
        {
          call Leds.redToggle();
          ok = call RadioSend.send(received);
        }
        else
          {
            call Leds.greenToggle();
            received->addr = TOS_UART_ADDR;
            ok = call UARTSend.send(received);
          }
    if (ok != FAIL)
        {
```

```
            dbg(DBG_USR1, "GenericBase send pending\n");
            sendPending = TRUE;
          }
        else {
            call Leds.yellowToggle();
        }

      }
      return nextReceiveBuffer;
    }

    result_t sendDone(TOS_MsgPtr sent, result_t success) {
      if(ourBuffer == sent)
        {
            dbg(DBG_USR1, "GenericBase send buffer free\n");
            if (success == FAIL)
              call Leds.yellowToggle();
            sendPending = FALSE;
        }
      return SUCCESS;
    }

    event TOS_MsgPtr RadioReceive.receive(TOS_MsgPtr data) {
      if (data->crc) {
        return receive(data, FALSE);
      }
      else {
        return data;
      }
    }

    event TOS_MsgPtr UARTReceive.receive(TOS_MsgPtr data) {
      return receive(data, TRUE);
    }

    event result_t UARTSend.sendDone(TOS_MsgPtr msg, result_t success) {
      return sendDone(msg, success);
    }

    event result_t RadioSend.sendDone(TOS_MsgPtr msg, result_t success) {
      return sendDone(msg, success);
    }
  }
```

A.3 Application Code for ListenRaw

File: ListenRaw.java

```
/*
 * This software is copyrighted by Mike Chen and the Regents of
 * the University of California.  The following terms apply to all
 * files associated with the software unless explicitly disclaimed in
 * individual files.
 *
 * The authors hereby grant permission to use this software without
 * fee or royalty for any non-commercial purpose.  The authors also
 * grant permission to redistribute this software, provided this
 * copyright and a copy of this license (for reference) are retained
 * in all distributed copies.
 *
 * For commercial use of this software, contact the authors.
 *
 * IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY
 * FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES
 * ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY
 * DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 *
 * THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT.  THIS SOFTWARE
 * IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE
 * NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR
 * MODIFICATIONS.
 *
 * Authors: Mike Chen, Philip Levis
 * Last Modified: 7/1/02 (transition to nesC)
 *
 */


package net.tinyos.tools;

import java.util.*;
import java.io.*;
import javax.comm.*;
import java.text.*;

import net.tinyos.util.*;

public class ListenRaw {

    private static String CLASS_NAME = "net.tinyos.tools.ListenRaw";
    private static final int MAX_MSG_SIZE = 36;
    private static final int PORT_SPEED = 19200;
    private static final int LENGTH_OFFSET = 4;
    private int packetLength;
```

```
    private static int counter =0;

    // Toggle with -e flag
    private static boolean showEntireMessage = false;

    private CommPortIdentifier portId;
    private SerialPort port;
    private String portName;
    private InputStream in;
    private OutputStream out;

    public ListenRaw(String portName) {
        this.portName = portName;
    }


    public void open() throws NoSuchPortException, PortInUseException, IOException,
UnsupportedCommOperationException {
        //System.out.println("Opening port " + portName);
      System.out.println("Number, Acceleration in g units, time, moteID");
        portId = CommPortIdentifier.getPortIdentifier(portName);
        port = (SerialPort)portId.open(CLASS_NAME, 0);
        in = port.getInputStream();
        out = port.getOutputStream();

        port.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);
        port.disableReceiveFraming();
        //printPortStatus();
        // These are the mote UART parameters
        port.setSerialPortParams(PORT_SPEED,
                                 SerialPort.DATABITS_8,
                                 SerialPort.STOPBITS_1,
                                 SerialPort.PARITY_NONE);
        //printPortStatus();
        //System.out.println();
    }

    private void printPortStatus() {
        System.out.println(" baud rate: " + port.getBaudRate());
        System.out.println(" data bits: " + port.getDataBits());
        System.out.println(" stop bits: " + port.getStopBits());
        System.out.println(" parity:    " + port.getParity());
    }

    private static void printAllPorts() {
        Enumeration ports = CommPortIdentifier.getPortIdentifiers();

        if (ports == null) {
            System.out.println("No comm ports found!");
            return;
        }

        // print out all ports
```

```
        System.out.println("printing all ports...");
        while (ports.hasMoreElements()) {
            System.out.println("  " + ((CommPortIdentifier)ports.nextElement()).getName());
        }
    }



    public void read() throws IOException {
        int i;
        int count = 0;
        int[] packet = new int[MAX_MSG_SIZE];

        while ((i = in.read()) != -1) {
            String val = Integer.toHexString( i &0xff);
            if (val.length() == 1) {
                val = "0" + val;
            }
            if(i == 0x7e || count != 0){
                packet[count] = i;
                if (count == LENGTH_OFFSET) { // Figure out length of packet
                    //System.out.print(val + " ");
                    packetLength = i + count;
                    if (packetLength > MAX_MSG_SIZE - LENGTH_OFFSET) {
                        System.err.print("!"); // If too long, print a !
                        packetLength = MAX_MSG_SIZE;
                    }
                }
                // Don't print data after the packet
                else if (!showEntireMessage && (count > packetLength) && (count <
MAX_MSG_SIZE)) {}
                else {
                  // System.out.print(val + " "); // Packet data
                }
                count++;

                if (count >= MAX_MSG_SIZE) {
                    //System.out.println();
            Calendar calendar = new GregorianCalendar();
            DecimalFormat myformat = new DecimalFormat();
                    myformat.setMaximumFractionDigits(5);
            double j ;
            int l = packet[4];
            for( int k = 12; k < 11 + l -6; k =k+2)
             {
               j = (packet[k] *256 + packet[k-1]);//- 538)/62.0;
              /*  System.out.print("time:  "  +  calendar.get(Calendar.HOUR_OF_DAY)  +  ":"  +
calendar.get(Calendar.MINUTE) +":"
                +calendar.get(Calendar.SECOND) + ":" + calendar.get(Calendar.MILLISECOND));
                System.out.println(   " id: " + (packet[6]*256 + packet[5]) + " acc: " + j + "g" +
"count is: " + (counter++));*/
                System.out.println("  "  +(counter++)+  "  ,  "  +    myformat.format(j)  +  ",  "
+calendar.get(Calendar.HOUR_OF_DAY) + ":" + calendar.get(Calendar.MINUTE) +":"
```

```
                        +calendar.get(Calendar.SECOND)          +          ":"          +
calendar.get(Calendar.MILLISECOND) + ", " +(packet[6]*256 + packet[5]));
             }
                    count = 0;
                    packetLength = MAX_MSG_SIZE;
                }
            }
            else{
                System.out.println("extra byte: " + val);
            }
        }
    }

    private static void printUsage() {
        System.err.println("usage: java listen [options] <port>");
        System.err.println("options are:");
        System.err.println("  -h, --help:    usage help");
        System.err.println("  -p:            print available ports");
        System.err.println("  -e:            display entire message");
        System.exit(-1);
    }


    public static void main(String args[]) {

     if ((args.length < 1) || (args.length > 2)) {
         printUsage();
     }

         for (int i = 0; i < args.length; i++) {
            if (args[i].equals("-h") || args[i].equals("--help")) {
                printUsage();
            }
            if (args[i].equals("-p")) {
                printAllPorts();
            }
            if (args[i].equals("-e")) {
                showEntireMessage = true;
            }
         }

         if (args[args.length - 1].charAt(0) == '-') {
            return; // No port specified
         }

         ListenRaw reader = new ListenRaw(args[args.length - 1]);
         try {
            reader.open();
         }
         catch (Exception e) {
            e.printStackTrace();
         }
```

```
        try {
            reader.read();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Appendix B

Data Sheets

New

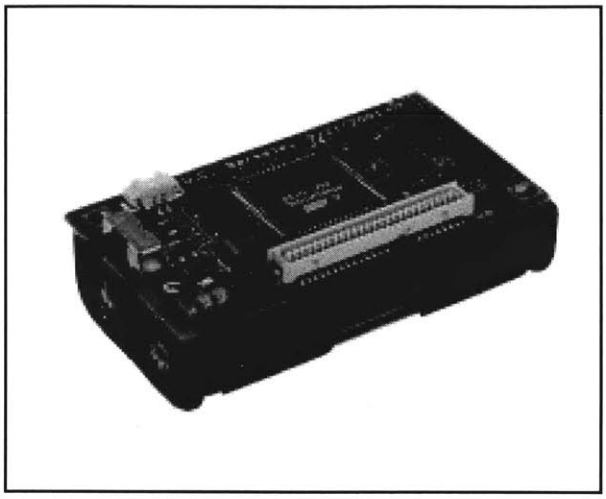
MICA

WIRELESS MEASUREMENT SYSTEM

- 2nd Generation, Tiny, Wireless Smart Sensors
- TinyOS - Unprecedented Communications and Processing
- AA/Year Battery Life
- Small Form Factor
- Wireless Communications
- Light, Temperature, Acceleration/Seismic, Acoustic, and Magnetic Sensors
- Developed for DARPA NEST Program

Applications

- Wireless Sensor Networks
- Security, Surveillance, and Force Protection
- Environmental Monitoring
- Large Scale Wireless Networks (1000+ points)
- Distributed Computing Platform



MICA

The MICA Mote is a second generation mote module used for research and development of low power, wireless, sensor networks. The MICA mote was developed by UC Berkeley's research group on wireless sensors. It consists of:

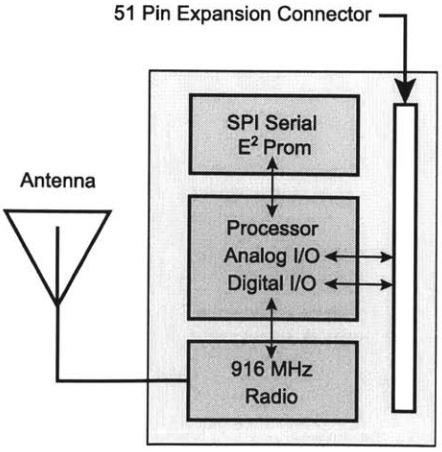
- Plug-in sensor boards
- TinyOS (TOS) Distributed Software Operating System.
- Atmega 128L processor
- 916MHz or 433MHz transceiver
- Attached AA(2) battery pack

TinyOS is a small, open-source, energy efficient, software operating system developed by UC Berkeley which supports large scale, self-configuring sensor networks. The source code and software development tools are publicly available at:

http://webs.cs.berkeley.edu/tos

TOS includes:

- Radio messaging
- Message hopping from mote to mote
- Low power modes
- Sensor measurements and signal processing



MPR300CA Block Diagram

Processor and Radio Platform (MPR300CB):
The MPR300CB is based on the Atmel ATmega 128L. The ATmega 128L is a low power microcontroller which runs TOS from its internal flash memory. The 128L has been selected for its low power and other features. The MPR300 (MICA) uses an ISM band radio transceiver module for wireless communication.

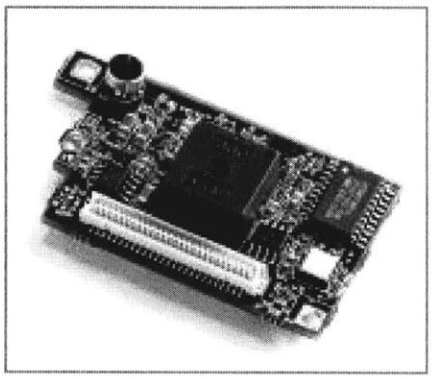
Sensor Boards:
Various sensor boards are available from Crossbow and other sources. These boards connect onto the MICA through a surface mount 51-pin connector. The 51-pin connector supports Analog Inputs, I2C, SPI, UART, and a multiplexed Address/Data bus. These interfaces make it easy to connect to a wide variety of external peripherals. Crossbow supplies the following sensor boards:

- MTS101CA Photo diode/ Thermistor/Proto and Experiment Board
- MTS300CA Photo diode, Thermistor, Microphone, and Sounder
- MTS310CA Same as MTS300CA but also including Magnetic and Acceleration Sensor

Processor/Radio Board	MPR300CB	Remarks
Speed	4MHz	
Flash	128K bytes	
SRAM	4K bytes	
EEPROM	4K bytes	
Serial Comms	UART	
A/D	10 bit ADC	8 channel
Processor Current Draw	5.5 mA	active current, typ
	<20uA	sleep mode, typ
Serial Flash	4Mbit	permanent ID 64 bits
Radio Frequency	916 MHz	ISM band (See Note)
Data Rate	40 Kbits/sec	max
Power	0.75 mW	
Radio Current Draw	12mA	transmit current, typ
	1.8 mA	receive current, typ
	<1uA	sleep current, typ
Radio Range	100 feet	programmable (See Note)
Power	2X AA batteries	attached pack
External Power	3 Volts	connector provided
User Interface	3 LEDs	user programmable
Expansion Connector	51 pin	connector for plug-in sensor boards

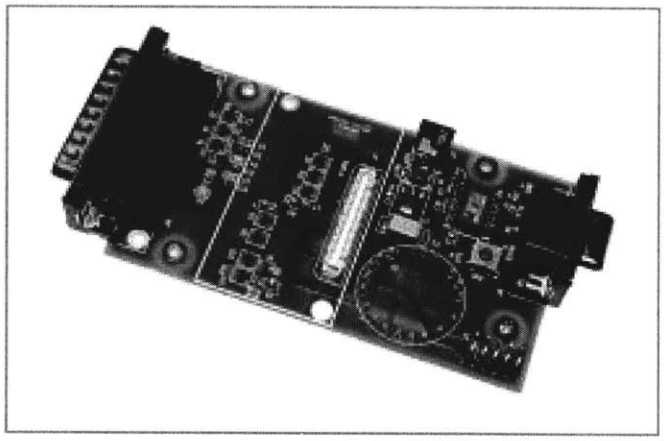
Notes: 433 MHz ISM band radio alternate is available - MPR310CA
Radio range dependent on antennae configuration. External antennae improves range.

▼ MTS310CA Sensor Board



Base Station/ Interface Board:
The base station allows the aggregation of sensor network data onto a PC or other computer platform. The MICA architecture is unique in that any sensor node (MPR300CB) module can function as a basestation by plugging the MPR300CB processor/radio board into a basic interface board, known as the Mote interface board.

▼ MIB500CA Mote Interface Board



Model	Description
MOTE-KIT301	Multipoint Developer's Kit (3X MPR300CB, 2X MTS300CA, 1X MIB300CA)
MOTE-KIT311	Advanced Multipoint Developer's Kit (4X MPR300CB, 3X MTS310CA, 1X MIB300CA)
MTS101CA	Light, Temp, and Prototype Sensor Board
MTS300CA	Light, Temp, Acoustic, and Sounder Sensor Board
MTS310CA	Same as MTS300CA but also includes Magnetic and Acceleration
MPR300CB	916 MHz Processor/Radio Board
MPR310CA	433 MHz Processor/Radio Board
MIB500CA	Mote Interface Board

Low Cost ±2 *g*/±10 *g* Dual Axis *i*MEMS® Accelerometers with Digital Output

ADXL202/ADXL210

FEATURES

2-Axis Acceleration Sensor on a Single IC Chip
Measures Static Acceleration as Well as Dynamic Acceleration
Duty Cycle Output with User Adjustable Period
Low Power <0.6 mA
Faster Response than Electrolytic, Mercury or Thermal Tilt Sensors
Bandwidth Adjustment with a Single Capacitor Per Axis
5 m*g* Resolution at 60 Hz Bandwidth
+3 V to +5.25 V Single Supply Operation
1000 *g* Shock Survival

APPLICATIONS

2-Axis Tilt Sensing
Computer Peripherals
Inertial Navigation
Seismic Monitoring
Vehicle Security Systems
Battery Powered Motion Sensing

GENERAL DESCRIPTION

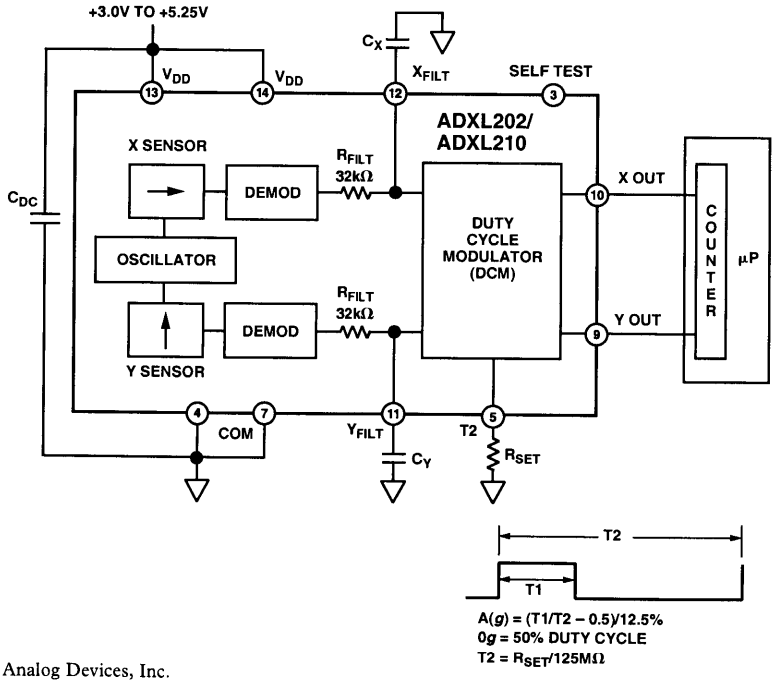
The ADXL202/ADXL210 are low cost, low power, complete 2-axis accelerometers with a measurement range of either ±2 *g*/±10 *g*. The ADXL202/ADXL210 can measure both dynamic acceleration (e.g., vibration) and static acceleration (e.g., gravity).

The outputs are digital signals whose duty cycles (ratio of pulsewidth to period) are proportional to the acceleration in each of the 2 sensitive axes. These outputs may be measured directly with a microprocessor counter, requiring no A/D converter or glue logic. The output period is adjustable from 0.5 ms to 10 ms via a single resistor (R_{SET}). If a voltage output is desired, a voltage output proportional to acceleration is available from the X_{FILT} and Y_{FILT} pins, or may be reconstructed by filtering the duty cycle outputs.

The bandwidth of the ADXL202/ADXL210 may be set from 0.01 Hz to 5 kHz via capacitors C_X and C_Y. The typical noise floor is 500 $\mu g/\sqrt{Hz}$ allowing signals below 5 m*g* to be resolved for bandwidths below 60 Hz.

The ADXL202/ADXL210 is available in a hermetic 14-lead Surface Mount CERPAK, specified over the 0°C to +70°C commercial or –40°C to +85°C industrial temperature range.

FUNCTIONAL BLOCK DIAGRAM



*i*MEMS is a registered trademark of Analog Devices, Inc.

REV. B

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781/329-4700 World Wide Web Site: http://www.analog.com
Fax: 781/326-8703

ADXL202/ADXL210–SPECIFICATIONS ($T_A = T_{MIN}$ to T_{MAX}, T_A = +25°C for J Grade only, V_{DD} = +5 V, R_{SET} = 125 kΩ, Acceleration = 0 *g*, unless otherwise noted)

Parameter	Conditions	ADXL202/JQC/AQC Min	Typ	Max	ADXL210/JQC/AQC Min	Typ	Max	Units
SENSOR INPUT	Each Axis							
Measurement Range[1]		±1.5	±2		±8	±10		*g*
Nonlinearity	Best Fit Straight Line		0.2			0.2		% of FS
Alignment Error[2]			±1			±1		Degrees
Alignment Error	X Sensor to Y Sensor		±0.01			±0.01		Degrees
Transverse Sensitivity[3]			±2			±2		%
SENSITIVITY	Each Axis							
Duty Cycle per *g*	T1/T2 @ +25°C	10	12.5	15	3.2	4.0	4.8	%/*g*
Sensitivity, Analog Output	At Pins X_{FILT}, Y_{FILT}		312			100		mV/*g*
Temperature Drift[4]	Δ from +25°C		±0.5			±0.5		% Rdg
ZERO *g* BIAS LEVEL	Each Axis							
0 *g* Duty Cycle	T1/T2	25	50	75	42	50	58	%
Initial Offset			±2			±2		*g*
0 *g* Duty Cycle vs. Supply			1.0	4.0		1.0	4.0	%/V
0 *g* Offset vs. Temperature[4]	Δ from +25°C		2.0			2.0		mg/°C
NOISE PERFORMANCE								
Noise Density[5]	@ +25°C		500	1000		500	1000	$\mu g/\sqrt{Hz}$
FREQUENCY RESPONSE								
3 dB Bandwidth	Duty Cycle Output		500			500		Hz
3 dB Bandwidth	At Pins X_{FILT}, Y_{FILT}		5			5		kHz
Sensor Resonant Frequency			10			14		kHz
FILTER								
R_{FILT} Tolerance	32 kΩ Nominal		±15			±15		%
Minimum Capacitance	At X_{FILT}, Y_{FILT}	1000			1000			pF
SELF TEST								
Duty Cycle Change	Self-Test "0" to "1"		10			10		%
DUTY CYCLE OUTPUT STAGE								
F_{SET}			125 MΩ/R_{SET}			125 MΩ/R_{SET}		
F_{SET} Tolerance	R_{SET} = 125 kΩ	0.7		1.3	0.7		1.3	kHz
Output High Voltage	I = 25 μA	V_S – 200 mV			V_S – 200 mV			mV
Output Low Voltage	I = 25 μA			200			200	mV
T2 Drift vs. Temperature			35			35		ppm/°C
Rise/Fall Time			200			200		ns
POWER SUPPLY								
Operating Voltage Range		3.0		5.25	2.7		5.25	V
Specified Performance		4.75		5.25	4.75		5.25	V
Quiescent Supply Current			0.6	1.0		0.6	1.0	mA
Turn-On Time[6]	To 99%		160 C_{FILT} + 0.3			160 C_{FILT} + 0.3		ms
TEMPERATURE RANGE								
Operating Range	JQC	0		+70	0		+70	°C
Specified Performance	AQC	–40		+85	–40		+85	°C

NOTES

[1]For all combinations of offset and sensitivity variation.

[2]Alignment error is specified as the angle between the true and indicated axis of sensitivity.

[3]Transverse sensitivity is the algebraic sum of the alignment and the inherent sensitivity errors.

[4]Specification refers to the maximum change in parameter from its initial at +25°C to its worst case value at T_{MIN} to T_{MAX}.

[5]Noise density ($\mu g/\sqrt{Hz}$) is the average noise at any frequency in the bandwidth of the part.

[6]C_{FILT} in μF. Addition of filter capacitor will increase turn on time. Please see the Application section on power cycling.

All min and max specifications are guaranteed. Typical specifications are not tested or guaranteed.

Specifications subject to change without notice.

ABSOLUTE MAXIMUM RATINGS*

Acceleration (Any Axis, Unpowered for 0.5 ms) ..... 1000 *g*
Acceleration (Any Axis, Powered for 0.5 ms) ..... 500 *g*
$+V_S$ ..... –0.3 V to +7.0 V
Output Short Circuit Duration
(Any Pin to Common) ..... Indefinite
Operating Temperature .....–55°C to +125°C
Storage Temperature .....–65°C to +150°C

*Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; the functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Drops onto hard surfaces can cause shocks of greater than 1000 *g* and exceed the absolute maximum rating of the device. Care should be exercised in handling to avoid damage.

PIN FUNCTION DESCRIPTIONS

Pin	Name	Description
1	NC	No Connect
2	V_{TP}	Test Point, Do Not Connect
3	ST	Self Test
4	COM	Common
5	T2	Connect R_{SET} to Set T2 Period
6	NC	No Connect
7	COM	Common
8	NC	No Connect
9	Y_{OUT}	Y Axis Duty Cycle Output
10	X_{OUT}	X Axis Duty Cycle Output
11	Y_{FILT}	Connect Capacitor for Y Filter
12	X_{FILT}	Connect Capacitor for X Filter
13	V_{DD}	+3 V to +5.25 V, Connect to 14
14	V_{DD}	+3 V to +5.25 V, Connect to 13

PACKAGE CHARACTERISTICS

Package	θ_{JA}	θ_{JC}	Device Weight
14-Lead CERPAK	110°C/W	30°C/W	5 Grams

PIN CONFIGURATION

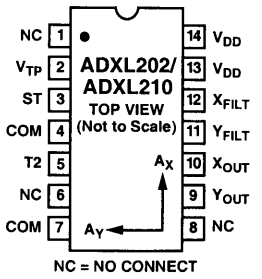


Figure 1 shows the response of the ADXL202 to the Earth's gravitational field. The output values shown are nominal. They are presented to show the user what type of response to expect from each of the output pins due to changes in orientation with respect to the Earth. The ADXL210 reacts similarly with output changes appropriate to its scale.

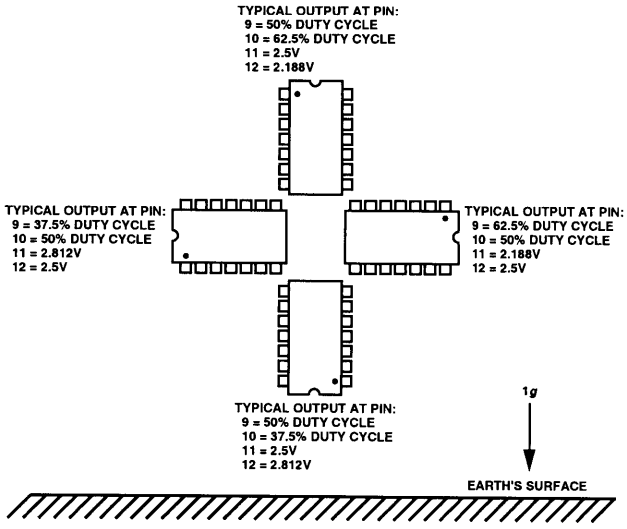


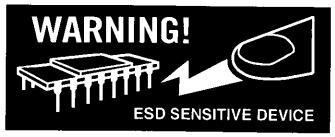
Figure 1. ADXL202/ADXL210 Nominal Response Due to Gravity

ORDERING GUIDE

Model	*g* Range	Temperature Range	Package Description	Package Option
ADXL202JQC	±2	0°C to +70°C	14-Lead CERPAK	QC-14
ADXL202AQC	±2	–40°C to +85°C	14-Lead CERPAK	QC-14
ADXL210JQC	±10	0°C to +70°C	14-Lead CERPAK	QC-14
ADXL210AQC	±10	–40°C to +85°C	14-Lead CERPAK	QC-14

CAUTION

ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although the ADXL202/ADXL210 features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.



TYPICAL CHARACTERISTICS (@ +25°C R_{SET} = 125 kΩ, V_{DD} = +5 V, unless otherwise noted)

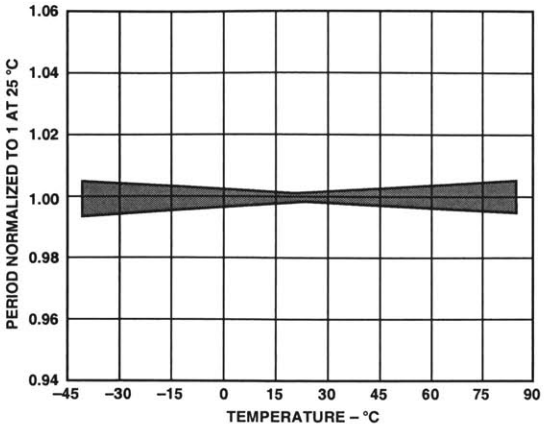


Figure 2. Normalized DCM Period (T2) vs. Temperature

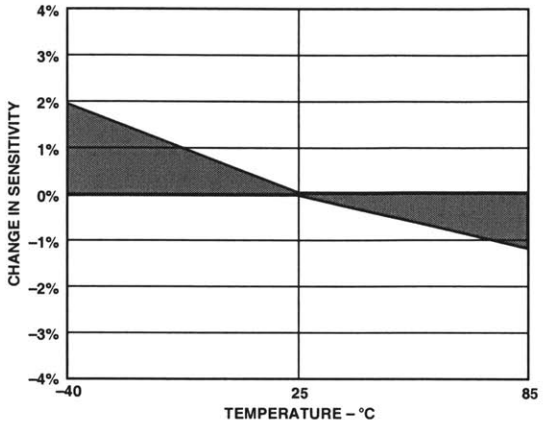


Figure 5. Typical X Axis Sensitivity Drift Due to Temperature

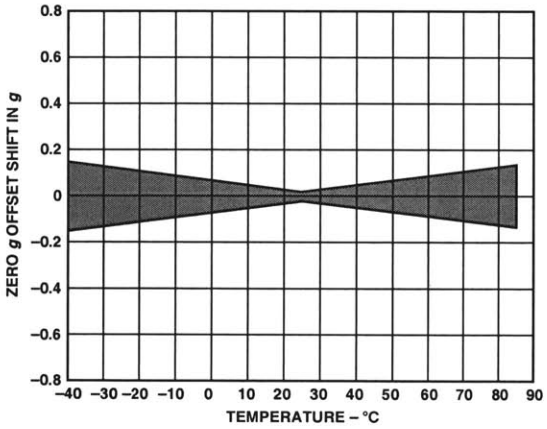


Figure 3. Typical Zero g Offset vs. Temperature

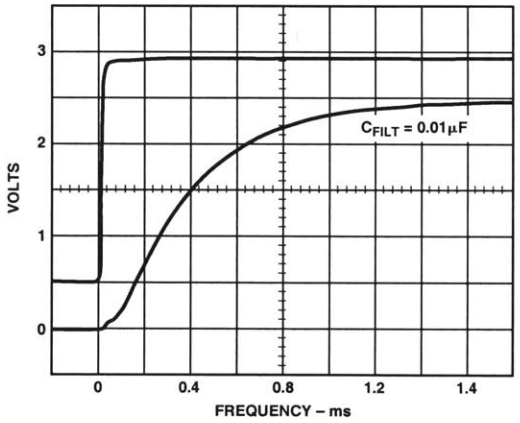


Figure 6. Typical Turn-On Time

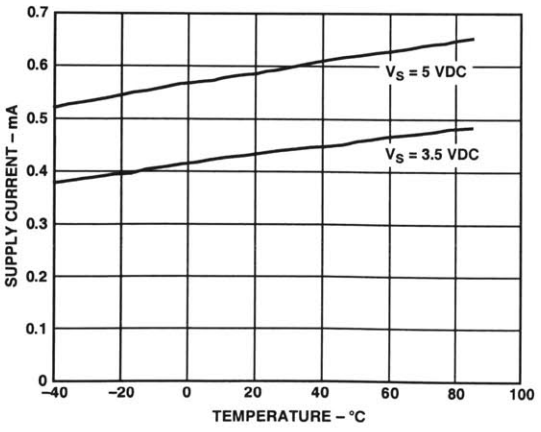


Figure 4. Typical Supply Current vs. Temperature

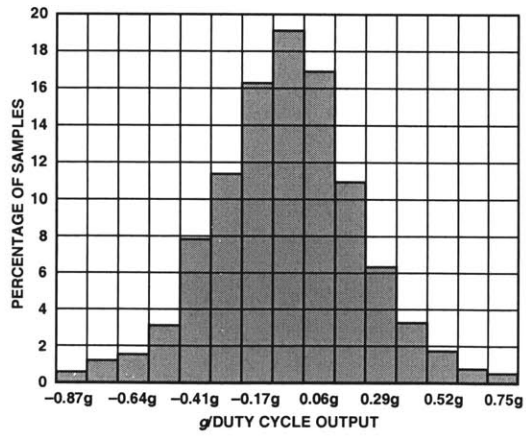


Figure 7. Typical Zero g Distribution at +25°C

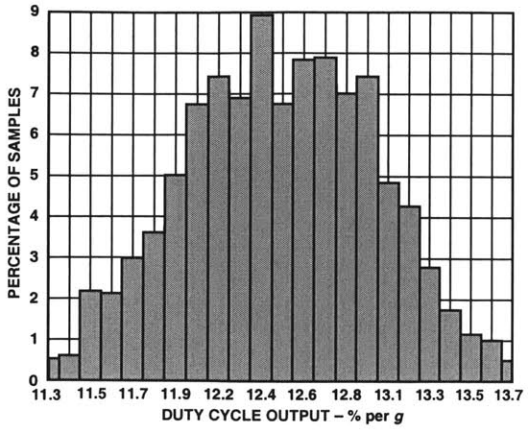


Figure 8. Typical Sensitivity per g at +25°C

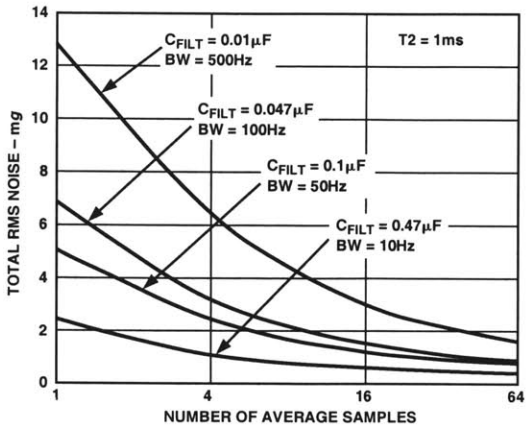


Figure 10. Typical Noise at Digital Outputs

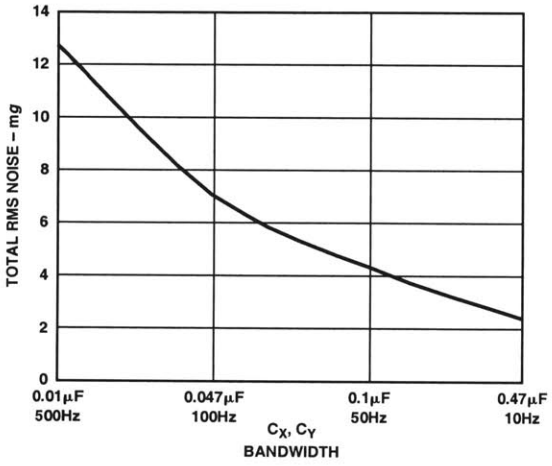


Figure 9. Typical Noise at X_{FILT} Output

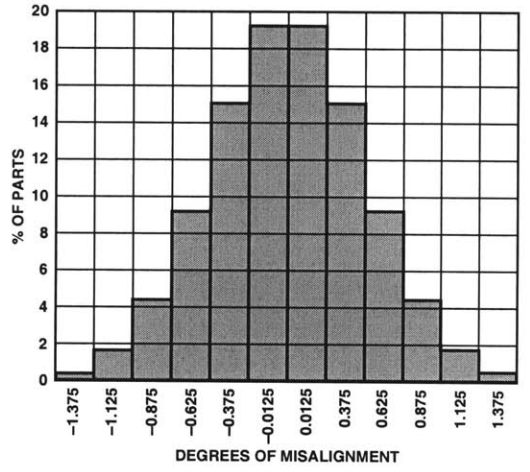


Figure 11. Rotational Die Alignment

DEFINITIONS

T1	Length of the "on" portion of the cycle.
T2	Length of the total cycle.
Duty Cycle	Ratio of the "on" time (T1) of the cycle to the total cycle (T2). Defined as T1/T2 for the ADXL202/ADXL210.
Pulsewidth	Time period of the "on" pulse. Defined as T1 for the ADXL202/ADXL210.

THEORY OF OPERATION

The ADXL202/ADXL210 are complete dual axis acceleration measurement systems on a single monolithic IC. They contain a polysilicon surface-micromachined sensor and signal conditioning circuitry to implement an open loop acceleration measurement architecture. For each axis, an output circuit converts the analog signal to a duty cycle modulated (DCM) digital signal that can be decoded with a counter/timer port on a microprocessor. The ADXL202/ADXL210 are capable of measuring both positive and negative accelerations to a maximum level of ±2 *g* or ±10 *g*. The accelerometer measures static acceleration forces such as gravity, allowing it to be used as a tilt sensor.

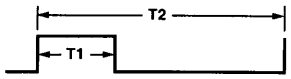
The sensor is a surface micromachined polysilicon structure built on top of the silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against acceleration forces. Deflection of the structure is measured using a differential capacitor that consists of independent fixed plates and central plates attached to the moving mass. The fixed plates are driven by 180° out of phase square waves. An acceleration will deflect the beam and unbalance the differential capacitor, resulting in an output square wave whose amplitude is proportional to acceleration. Phase sensitive demodulation techniques are then used to rectify the signal and determine the direction of the acceleration.

The output of the demodulator drives a duty cycle modulator (DCM) stage through a 32 kΩ resistor. At this point a pin is available on each channel to allow the user to set the signal bandwidth of the device by adding a capacitor. This filtering improves measurement resolution and helps prevent aliasing.

After being low-pass filtered, the analog signal is converted to a duty cycle modulated signal by the DCM stage. A single resistor sets the period for a complete cycle (T2), which can be set between 0.5 ms and 10 ms (see Figure 12). A 0 *g* acceleration produces a nominally 50% duty cycle. The acceleration signal can be determined by measuring the length of the T1 and T2 pulses with a counter/timer or with a polling loop using a low cost microcontroller.

An analog output voltage can be obtained either by buffering the signal from the X_{FILT} and Y_{FILT} pin, or by passing the duty cycle signal through an RC filter to reconstruct the dc value.

The ADXL202/ADXL210 will operate with supply voltages as low as 3.0 V or as high as 5.25 V.



A(*g*) = (T1/T2 – 0.5)/12.5%
0*g* = 50% DUTY CYCLE
T2(s) = R_{SET}(Ω)/125MΩ

Figure 12. Typical Output Duty Cycle

APPLICATIONS

POWER SUPPLY DECOUPLING

For most applications a single 0.1 µF capacitor, C_{DC}, will adequately decouple the accelerometer from signal and noise on the power supply. However, in some cases, especially where digital devices such as microcontrollers share the same power supply, digital noise on the supply may cause interference on the ADXL202/ADXL210 output. This is often observed as a slowly undulating fluctuation of voltage at X_{FILT} and Y_{FILT}. If additional decoupling is needed, a 100 Ω (or smaller) resistor or ferrite beads, may be inserted in the ADXL202/ADXL210's supply line.

DESIGN PROCEDURE FOR THE ADXL202/ADXL210

The design procedure for using the ADXL202/ADXL210 with a duty cycle output involves selecting a duty cycle period and a filter capacitor. A proper design will take into account the application requirements for bandwidth, signal resolution and acquisition time, as discussed in the following sections.

V_{DD}

The ADXL202/ADXL210 have two power supply (V_{DD}) Pins: 13 and 14. These two pins should be connected directly together.

COM

The ADXL202/ADXL210 have two commons, Pins 4 and 7. These two pins should be connected directly together and Pin 7 grounded.

V_{TP}

This pin is to be left open; make no connections of any kind to this pin.

Decoupling Capacitor C_{DC}

A 0.1 µF capacitor is recommended from V_{DD} to COM for power supply decoupling.

ST

The ST pin controls the self-test feature. When this pin is set to V_{DD}, an electrostatic force is exerted on the beam of the accelerometer. The resulting movement of the beam allows the user to test if the accelerometer is functional. The typical change in output will be 10% at the duty cycle outputs (corresponding to 800 m*g*). This pin may be left open circuit or connected to common in normal use.

Duty Cycle Decoding

The ADXL202/ADXL210's digital output is a duty cycle modulator. Acceleration is proportional to the ratio T1/T2. The nominal output of the ADXL202 is:

0 *g* = 50% *Duty Cycle*

Scale factor is 12.5% Duty Cycle Change per *g*

The nominal output of the ADXL210 is:

0 *g* = 50% *Duty Cycle*

Scale factor is 4% Duty Cycle Change per *g*

These nominal values are affected by the initial tolerance of the device including zero *g* offset error and sensitivity error.

T2 does not have to be measured for every measurement cycle. It need only be updated to account for changes due to temperature, (a relatively slow process). Since the T2 time period is shared by both X and Y channels, it is necessary only to measure it on one channel of the ADXL202/ADXL210. Decoding algorithms for various microcontrollers have been developed. Consult the appropriate Application Note.

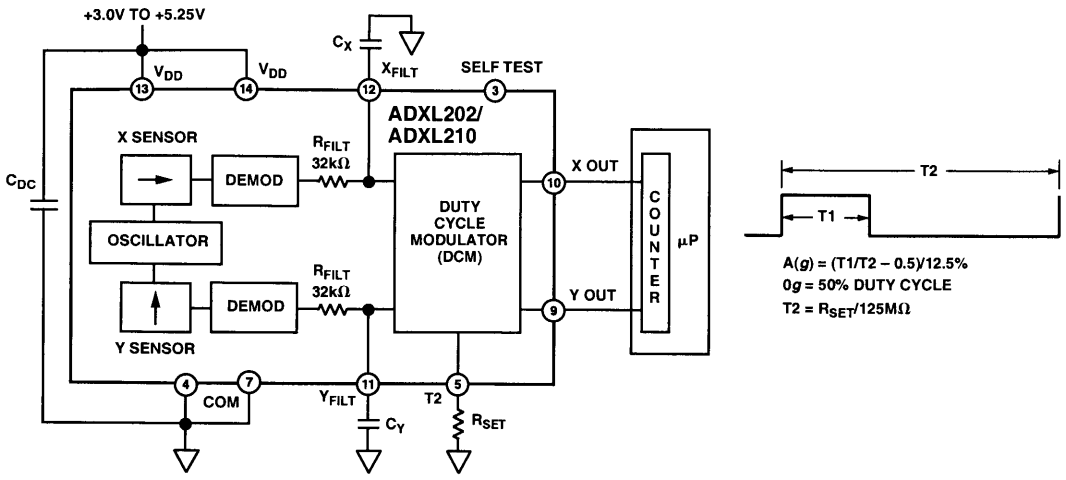


Figure 13. Block Diagram

Setting the Bandwidth Using C_X and C_Y

The ADXL202/ADXL210 have provisions for bandlimiting the X_{FILT} and Y_{FILT} pins. Capacitors must be added at these pins to implement low-pass filtering for antialiasing and noise reduction. The equation for the 3 dB bandwidth is:

$$F_{-3\,dB} = \frac{1}{\left(2\,\pi\,(32\,k\Omega) \times C(x,y)\right)}$$

or, more simply, $F_{-3\,dB} = \frac{5\,\mu F}{C_{(X,Y)}}$

The tolerance of the internal resistor (R_{FILT}), can vary as much as ±25% of its nominal value of 32 kΩ; so the bandwidth will vary accordingly. A minimum capacitance of 1000 pF for $C_{(X, Y)}$ is required in all cases.

Table I. Filter Capacitor Selection, C_X and C_Y

Bandwidth	Capacitor Value
10 Hz	0.47 μF
50 Hz	0.10 μF
100 Hz	0.05 μF
200 Hz	0.027 μF
500 Hz	0.01 μF
5 kHz	0.001 μF

Setting the DCM Period with R_{SET}

The period of the DCM output is set for both channels by a single resistor from R_{SET} to ground. The equation for the period is:

$$T2 = \frac{R_{SET}\,(\Omega)}{125\,M\Omega}$$

A 125 kΩ resistor will set the duty cycle repetition rate to approximately 1 kHz, or 1 ms. The device is designed to operate at duty cycle periods between 0.5 ms and 10 ms.

Table II. Resistor Values to Set T2

T2	R_{SET}
1 ms	125 kΩ
2 ms	250 kΩ
5 ms	625 kΩ
10 ms	1.25 MΩ

Note that the R_{SET} should always be included, even if only an analog output is desired. Use an R_{SET} value between 500 kΩ and 2 MΩ when taking the output from X_{FILT} or Y_{FILT}. The R_{SET} resistor should be place close to the T2 Pin to minimize parasitic capacitance at this node.

Selecting the Right Accelerometer

For most tilt sensing applications the ADXL202 is the most appropriate accelerometer. Its higher sensitivity (12.5%/*g* allows the user to use a lower speed counter for PWM decoding while maintaining high resolution. The ADXL210 should be used in applications where accelerations of greater than ±2 *g* are expected.

MICROCOMPUTER INTERFACES

The ADXL202/ADXL210 were specifically designed to work with low cost microcontrollers. Specific code sets, reference designs, and application notes are available from the factory. This section will outline a general design procedure and discuss the various trade-offs that need to be considered.

The designer should have some idea of the required performance of the system in terms of:

Resolution: the smallest signal change that needs to be detected.

Bandwidth: the highest frequency that needs to be detected.

Acquisition Time: the time that will be available to acquire the signal on each axis.

These requirements will help to determine the accelerometer bandwidth, the speed of the microcontroller clock and the length of the T2 period.

When selecting a microcontroller it is helpful to have a counter timer port available. The microcontroller should have provisions for software calibration. While the ADXL202/ADXL210 are highly accurate accelerometers, they have a wide tolerance for

initial offset. The easiest way to null this offset is with a calibration factor saved on the microcontroller or by a user calibration for zero *g*. In the case where the offset is calibrated during manufacture, there are several options, including external EEPROM and microcontrollers with "one-time programmable" features.

DESIGN TRADE-OFFS FOR SELECTING FILTER CHARACTERISTICS: THE NOISE/BW TRADE-OFF

The accelerometer bandwidth selected will determine the measurement resolution (smallest detectable acceleration). Filtering can be used to lower the noise floor and improve the resolution of the accelerometer. Resolution is dependent on both the analog filter bandwidth at X_{FILT} and Y_{FILT} and on the speed of the microcontroller counter.

The analog output of the ADXL202/ADXL210 has a typical bandwidth of 5 kHz, much higher than the duty cycle stage is capable of converting. The user must filter the signal at this point to limit aliasing errors. To minimize DCM errors the analog bandwidth should be less than 1/10 the DCM frequency. Analog bandwidth may be increased to up to 1/2 the DCM frequency in many applications. This will result in greater dynamic error generated at the DCM.

The analog bandwidth may be further decreased to reduce noise and improve resolution. The ADXL202/ADXL210 noise has the characteristics of white Gaussian noise that contributes equally at all frequencies and is described in terms of μg per root Hz; i.e., the noise is proportional to the square root of the bandwidth of the accelerometer. It is recommended that the user limit bandwidth to the lowest frequency needed by the application, to maximize the resolution and dynamic range of the accelerometer.

With the single pole roll-off characteristic, the typical noise of the ADXL202/ADXL210 is determined by the following equation:

$$Noise\ (rms) = \left(500\ \mu g/\sqrt{Hz}\right) \times \left(\sqrt{BW \times 1.5}\right)$$

At 100 Hz the noise will be:

$$Noise\ (rms) = \left(500\ \mu g/\sqrt{Hz}\right) \times \left(\sqrt{100 \times (1.5)}\right) = 6.12\ mg$$

Often the peak value of the noise is desired. Peak-to-peak noise can only be estimated by statistical methods. Table III is useful for estimating the probabilities of exceeding various peak values, given the rms value.

Table III. Estimation of Peak-to-Peak Noise

Nominal Peak-to-Peak Value	% of Time that Noise Will Exceed Nominal Peak-to-Peak Value
2.0 × rms	32%
4.0 × rms	4.6%
6.0 × rms	0.27%
8.0 × rms	0.006%

The peak-to-peak noise value will give the best estimate of the uncertainty in a single measurement.

Table IV gives typical noise output of the ADXL202/ADXL210 for various C_X and C_Y values.

Table IV. Filter Capacitor Selection, C_X and C_Y

Bandwidth	C_X, C_Y	rms Noise	Peak-to-Peak Noise Estimate 95% Probability (rms × 4)
10 Hz	0.47 μF	1.9 *mg*	7.6 *mg*
50 Hz	0.10 μF	4.3 *mg*	17.2 *mg*
100 Hz	0.05 μF	6.1 *mg*	24.4 *mg*
200 Hz	0.027 μF	8.7 *mg*	35.8 *mg*
500 Hz	0.01 μF	13.7 *mg*	54.8 *mg*

CHOOSING T2 AND COUNTER FREQUENCY: DESIGN TRADE-OFFS

The noise level is one determinant of accelerometer resolution. The second relates to the measurement resolution of the counter when decoding the duty cycle output.

The ADXL202/ADXL210's duty cycle converter has a resolution of approximately 14 bits; better resolution than the accelerometer itself. The actual resolution of the acceleration signal is, however, limited by the time resolution of the counting devices used to decode the duty cycle. The faster the counter clock, the higher the resolution of the duty cycle and the shorter the T2 period can be for a given resolution. The following table shows some of the trade-offs. It is important to note that this is the resolution due to the microprocessors's counter. It is probable that the accelerometer's noise floor may set the lower limit on the resolution, as discussed in the previous section.

Table V. Trade-Offs Between Microcontroller Counter Rate, T2 Period and Resolution of Duty Cycle Modulator

T2 (ms)	R_{SET} (kΩ)	ADXL202/ ADXL210 Sample Rate	Counter-Clock Rate (MHz)	Counts per T2 Cycle	Counts per *g*	Resolution (mg)
1.0	124	1000	2.0	2000	250	4.0
1.0	124	1000	1.0	1000	125	8.0
1.0	124	1000	0.5	500	62.5	16.0
5.0	625	200	2.0	10000	1250	0.8
5.0	625	200	1.0	5000	625	1.6
5.0	625	200	0.5	2500	312.5	3.2
10.0	1250	100	2.0	20000	2500	0.4
10.0	1250	100	1.0	10000	1250	0.8
10.0	1250	100	0.5	5000	625	1.6

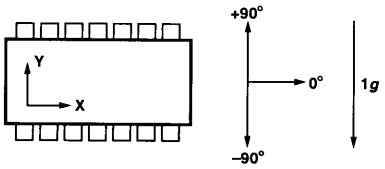
STRATEGIES FOR USING THE DUTY CYCLE OUTPUT WITH MICROCONTROLLERS

Application notes outlining various strategies for using the duty cycle output with low cost microcontrollers are available from the factory.

USING THE ADXL202/ADXL210 AS A DUAL AXIS TILT SENSOR

One of the most popular applications of the ADXL202/ADXL210 is tilt measurement. An accelerometer uses the force of gravity as an input vector to determine orientation of an object in space.

An accelerometer is most sensitive to tilt when its sensitive axis is perpendicular to the force of gravity, i.e., parallel to the earth's surface. At this orientation its sensitivity to changes in tilt is highest. When the accelerometer is oriented on axis to gravity, i.e., near its +1 *g* or –1 *g* reading, the change in output acceleration per degree of tilt is negligible. When the accelerometer is perpendicular to gravity, its output will change nearly 17.5 m*g* per degree of tilt, but at 45° degrees it is changing only at 12.2 m*g* per degree and resolution declines. The following table illustrates the changes in the X and Y axes as the device is tilted ±90° through gravity.



X AXIS ORIENTATION TO HORIZON (°)	X OUTPUT		Y OUTPUT (*g*)	
	X OUTPUT (*g*)	Δ PER DEGREE OF TILT (m*g*)	Y OUTPUT (*g*)	Δ PER DEGREE OF TILT (m*g*)
–90	–1.000	–0.2	0.000	17.5
–75	–0.966	4.4	0.259	16.9
–60	–0.866	8.6	0.500	15.2
–45	–0.707	12.2	0.707	12.4
–30	–0.500	15.0	0.866	8.9
–15	–0.259	16.8	0.966	4.7
0	0.000	17.5	1.000	0.2
15	0.259	16.9	0.966	–4.4
30	0.500	15.2	0.866	–8.6
45	0.707	12.4	0.707	–12.2
60	0.866	8.9	0.500	–15.0
75	0.966	4.7	0.259	–16.8
90	1.000	0.2	0.000	–17.5

Figure 14. How the X and Y Axes Respond to Changes in Tilt

A DUAL AXIS TILT SENSOR: CONVERTING ACCELERATION TO TILT

When the accelerometer is oriented so both its X and Y axes are parallel to the earth's surface it can be used as a two axis tilt sensor with a roll and a pitch axis. Once the output signal from the accelerometer has been converted to an acceleration that varies between –1 *g* and +1 *g*, the output tilt in degrees is calculated as follows:

$$Pitch = ASIN\ (Ax/1\ g)$$
$$Roll = ASIN\ (Ay/1\ g)$$

Be sure to account for overranges. It is possible for the accelerometers to output a signal greater than ±1 *g* due to vibration, shock or other accelerations.

MEASURING 360° OF TILT

It is possible to measure a full 360° of orientation through gravity by using two accelerometers oriented perpendicular to one another (see Figure 15). When one sensor is reading a maximum change in output per degree, the other is at its minimum.

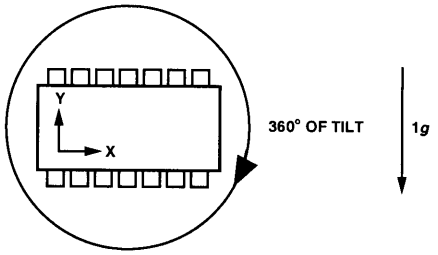


Figure 15. Using a Two-Axis Accelerometer to Measure 360° of Tilt

USING THE ANALOG OUTPUT

The ADXL202/ADXL210 was specifically designed for use with its digital outputs, but has provisions to provide analog outputs as well.

Duty Cycle Filtering

An analog output can be reconstructed by filtering the duty cycle output. This technique requires only passive components. The duty cycle period (T2) should be set to 1 ms. An RC filter with a 3 dB point at least a factor of 10 less than the duty cycle frequency is connected to the duty cycle output. The filter resistor should be no less than 100 kΩ to prevent loading of the output stage. The analog output signal will be ratiometric to the supply voltage. The advantage of this method is an output scale factor of approximately double the analog output. Its disadvantage is that the frequency response will be lower than when using the X_{FILT}, Y_{FILT} output.

X_{FILT}, Y_{FILT} Output

The second method is to use the analog output present at the X_{FILT} and Y_{FILT} pin. Unfortunately, these pins have a 32 kΩ output impedance and are not designed to drive a load directly. An op amp follower may be required to buffer this pin. The advantage of this method is that the full 5 kHz bandwidth of the accelerometer is available to the user. A capacitor still must be added at this point for filtering. The duty cycle converter should be kept running by using R_{SET} <10 MΩ. Note that the accelerometer offset and sensitivity are ratiometric to the supply voltage. The offset and sensitivity are nominally:

0 *g* Offset = $V_{DD}/2$	2.5 V at +5 V
ADXL202 Sensitivity = (60 mV × V_S)/*g*	300 mV/*g* at +5 V, V_{DD}
ADXL210 Sensitivity = (20 mV × V_S)/*g*	100 mV/*g* at +5 V, V_{DD}

USING THE ADXL202/ADXL210 IN VERY LOW POWER APPLICATIONS

An application note outlining low power strategies for the ADXL202/ADXL210 is available. Some key points are presented here. It is possible to reduce the ADXL202/ADXL210's average current from 0.6 mA to less than 20 μA by using the following techniques:

1. Power Cycle the accelerometer.
2. Run the accelerometer at a Lower Voltage, (Down to 3 V).

Power Cycling with an External A/D

Depending on the value of the X_{FILT} capacitor, the ADXL202/ADXL210 is capable of turning on and giving a good reading in 1.6 ms. Most microcontroller based A/Ds can acquire a reading in another 25 μs. Thus it is possible to turn on the ADXL202/ADXL210 and take a reading in <2 ms. If we assume that a 20 Hz sample rate is sufficient, the total current required to take 20 samples is 2 ms × 20 samples/s × 0.6 mA = 24 μA average current. Running the part at 3 V will reduce the supply current from 0.6 mA to 0.4 mA, bringing the average current down to 16 μA.

The A/D should read the analog output of the ADXL202/ADXL210 at the X_{FILT} and Y_{FILT} pins. A buffer amplifier is recommended, and may be required in any case to amplify the analog output to give enough resolution with an 8-bit to 10-bit converter.

Power Cycling When Using the Digital Output

An alternative is to run the microcontroller at a higher clock rate and put it into shutdown between readings, allowing the use of the digital output. In this approach the ADXL202/ADXL210 should be set at its fastest sample rate (T2 = 0.5 ms), with a 500 Hz filter at X_{FILT} and Y_{FILT}. The concept is to acquire a reading as quickly as possible and then shut down the ADXL202/ADXL210 and the microcontroller until the next sample is needed.

In either of the above approaches, the ADXL202/ADXL210 can be turned on and off directly using a digital port pin on the microcontroller to power the accelerometer without additional components. The port should be used to switch the common pin of the accelerometer so the port pin is "pulling down."

CALIBRATING THE ADXL202/ADXL210

The initial value of the offset and scale factor for the ADXL202/ADXL210 will require calibration for applications such as tilt measurement. The ADXL202/ADXL210 architecture has been designed so that these calibrations take place in the software of the microcontroller used to decode the duty cycle signal. Calibration factors can be stored in EEPROM or determined at turn-on and saved in dynamic memory.

For low *g* applications, the force of gravity is the most stable, accurate and convenient acceleration reference available. A reading of the 0 *g* point can be determined by orientating the device parallel to the earth's surface and then reading the output.

A more accurate calibration method is to make a measurements at +1 *g* and –1 *g*. The sensitivity can be determined by the two measurements.

To calibrate, the accelerometer's measurement axis is pointed directly at the earth. The 1 *g* reading is saved and the sensor is turned 180° to measure –1 *g*. Using the two readings, the sensitivity is:

Let A = Accelerometer output with axis oriented to +1 *g*
Let B = Accelerometer output with axis oriented to –1 *g then*:
Sensitivity = [A – *B*]/2 *g*

For example, if the +1 *g* reading (A) is 55% duty cycle and the –1 *g* reading (B) is 32% duty cycle, then:

Sensitivity = [55% – 32%]/2 *g* = 11.5%/*g*

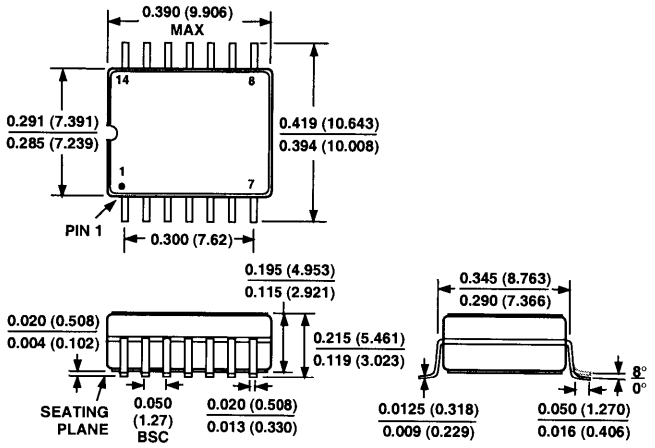
These equations apply whether the output is analog, or duty cycle.

Application notes outlining algorithms for calculating acceleration from duty cycle and automated calibration routines are available from the factory.

OUTLINE DIMENSIONS

Dimensions shown in inches and (mm).

14-Lead CERPAK
(QC-14)



C3037b–2–4/99

PRINTED IN U.S.A.